PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2001-209641
(43)Date of publication of application : 03.08.2001

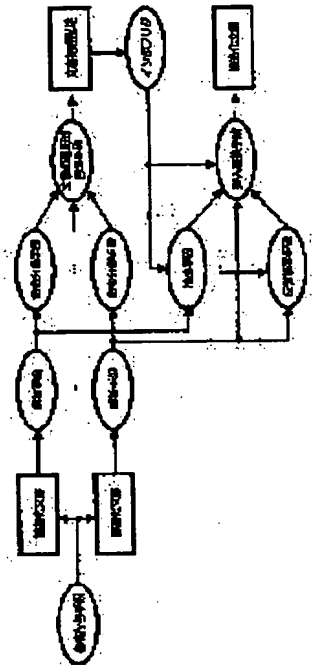
(51)Int.Cl. G06F 17/27

(21)Application number : 2000-018059
(22)Date of filing : 25.01.2000

(71)Applicant : FUJI XEROX CO LTD
(72)Inventor : KUNITAKE SETSU
YAMASHITA ICHIRO
KAWABE SHIGEHISA

(54) SYSTEM AND METHOD FOR PROCESSING STRUCTURED DOCUMENT

(57)Abstract:
PROBLEM TO BE SOLVED: To perform a document synthesizing processing by extracting document components from a structure document and inserting/replacing the respective document components in a model document without using a script with described procedure.
SOLUTION: In a structured document, an extraction instruction for taking out the document components and repetitive copying and insertion/replacement instructions are imparted. Thus, as the result of specifying the take-out of the document components, the repetitive copying and the document components (parts) for inserting or replacing the document components, dynamically synthesizing the instructions taken out from the inputted plural structured documents and preparing a document processing description, the need of a document processing description script is eliminated. Thus, the time and labor of managing the script separately from original documents are omitted.



LEGAL STATUS

[Date of request for examination] 09.01.2004

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's

decision of rejection]
[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号
特開2001－209641
(P2001－209641A)
(43)公開日　平成13年８月３日(2001.8.3)

(51)Int.Cl.⁷	識別記号	ＦＩ	テーマコード（参考）
Ｇ０６Ｆ　17/27		Ｃ０６Ｆ　15/20　５５０Ｅ	５Ｂ００９

審査請求　未請求　請求項の数14　ＯＬ　（全 47 頁）

(21)出願番号　特願2000－18059(P2000－18059)

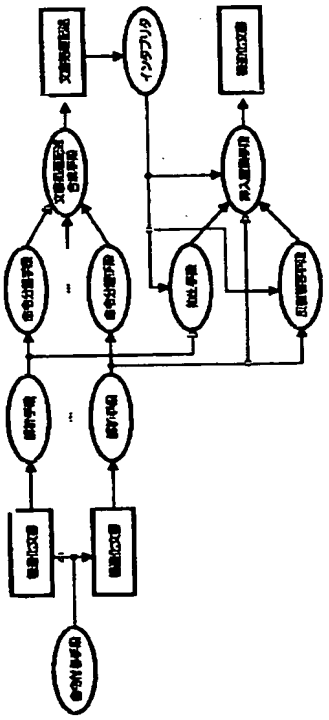
(22)出願日　平成12年１月25日(2000.1.25)

(71)出願人　000005496
富士ゼロックス株式会社
東京都港区赤坂二丁目17番22号
(72)発明者　國武　節
神奈川県足柄上郡中井町境430　グリーンテクなかい　富士ゼロックス株式会社内
(72)発明者　山下　一郎
神奈川県足柄上郡中井町境430　グリーンテクなかい　富士ゼロックス株式会社内
(74)代理人　100086531
弁理士　澤田　俊夫

最終頁に続く

(54)【発明の名称】　構造化文書処理システム及び構造化文書処理方法

(57)【要約】
【課題】　手続きを記述したスクリプトを用いることなしに、構造文書から文書部品を抽出し、各文書部品を雛型文書中に挿入・置換して、文書合成処理を行う。
【解決手段】　構造化文書中には、文書部品を取り出す抽出命令と、反復複写と挿入置換命令が付与されている。したがって、文書部品の取り出しの指定と、反復複写の指定と、文書部品を挿入または置換する文書部品（個所）の指定を行ない、入力した複数の構造化文書から取り出した命令を動的に合成して文書処理記述を作成するようにした結果、文書処理記述スクリプトが不要となる。したがって、元の文書と別にスクリプトを管理するという手間が省略される。



【特許請求の範囲】
【請求項１】構造的に表現され１以上の文書部品を含んだ構造化文書を処理するための構造化文書処理システムであって、
文書部品の抽出を指定する抽出命令又は文書部品の所定回数の複写を指定する反復複写命令又は文書部品の挿入若しくは置換を指定する挿入置換命令を該当する文書部品に付与する命令付与手段と、
構造化文書の構造を解析して構文解析木を生成する解析手段と、
前記解析手段により生成された構文解析木を基に構造化文書に付与された命令と文書部品を分離して命令を取り出す命令分離手段と、
抽出命令が付与された第１の構造化文書から前記命令分離手段によって取り出された抽出命令と、反復複写命令及び／又は挿入置換命令が付与された第２の構造化文書から前記命令分離手段によって取り出された反復複写命令及び／又は挿入置換命令とを併合・整列して、該第１及び第２の構造化文書を処理するための命令列からなる文書処理記述を生成する文書処理記述生成手段と、
抽出命令の対象である第１の文書部品を第１の構造化文書から取り出す抽出手段と、
反復複写命令の対象である文書部品と該文書部品付与された命令を所定回数だけ反復複写する反復複写手段と、
第１の文書部品を、挿入置換命令の対象である第２の文書部品の前又は後に挿入するか又は該第２の文書部品と置換する挿入置換手段と、
文書処理記述を順次解釈して、前記抽出手段、前記反復複写手段、及び／又は前記挿入置換手段を用いて文書部品を合成するインタプリタと、を具備することを特徴とする構造化文書処理システム。
【請求項２】前記抽出命令は文書部品の属性の取り出しを指定する属性抽出命令であり、
前記挿入置換命令は文書部品の属性の置換を指定する属性置換命令であり、
前記命令分離手段は、構造化文書から属性抽出命令と属性置換命令を取り出し、
前記挿入置換手段は、第１の文書部品の属性を属性置換命令の対象である第２の文書部品の属性と置換する属性置換手段であり、
前記インタプリタは、前記文書処理記述合成手段が合成した文書処理記述を解釈して、属性抽出命令で指定された文書部品の所定のノードの属性を抽出するとともに、該抽出された属性を属性置換命令で指定された文書部品の所定のノードに設定する、ことを特徴とする請求項１に記載の構造化文書処理システム。
【請求項３】前記属性置換手段は、属性置換命令に対して、文書部品にあらかじめ設定されている属性値文字列とシステムの状態から得られる文字列を組み合わせて合成される文字列に置き換える属性置換手段であることを特徴とする請求項２に記載の構造化文書処理システム。
【請求項４】前記抽出命令はパス名付き抽出命令であり、
前記反復複写命令及び挿入置換命令はそれぞれパターン式付き反復複写命令及びパターン式付き挿入置換命令であり、
前記命令分離手段は、構造化文書から、抽出命令とパス名を取り出すか、又は、反復複写命令と挿入置換命令とパターン式を取り出し、
前記反復複写手段は、抽出した文書部品に付与されたパス名とパターン式とのパターン・マッチングを行い、マッチするパス名を持つ文書部品の個数だけ反復複写を実行し、
前記挿入置換手段は、抽出した文書部品に付与されたパス名とパターン式とのパターン・マッチングを行い、マッチするパス名を持つ文書部品を挿入又は置換する、ことを特徴とする請求項１に記載の構造化文書処理システム。
【請求項５】前記挿入置換手段は、前記抽出手段によって取り出された文書部品に付与された抽出命令を挿入又は置換することを特徴とする請求項１に記載の構造化文書処理システム。
【請求項６】前記挿入置換手段は、前記抽出手段によって取り出された文書部品に付与された抽出命令を挿入又は置換する際に、パス名付き抽出命令のパス名を変更してから挿入置換することを特徴とする請求項６に記載の構造化文書処理システム。
【請求項７】ネットワーク接続された２以上のコンピュータで構成される分散ネットワーク・システム上におけるコンピュータ通信による協調的な処理によって実現される、構造的に表現され１以上の文書部品を含んだ構造化文書処理システムであって、
構造化文書を所定フォーマットのファイルとして格納するとともに、ファイル名を受信したことに応答して該当するファイルを前記ネットワーク経由で送信するファイル・サーバと、ファイルに対して文書処理を行う構造化文書処理サーバとを少なくとも含み、前記構造化文書処理サーバは、
文書部品の抽出を指定した抽出命令が付与された第１の構造化文書のファイル名と反復複写命令又は挿入置換命令を付与した第２の構造化文書のファイル名を含む処理起動記述を入力し解析して、該処理起動記述に含まれるファイル名を前記ファイル・サーバに前記ネットワーク経由で送信し、ファイル名に該当する各ファイルを前記ファイル・サーバからネットワーク経由で入力する入力手段と、
第１の構造化文書及び第２の構造化文書を解析して構文解析木を生成し、構文解析木を探索して文書部品と命令とを分離して命令を取り出し、各命令を併合・整列して構造化文書を処理するための命令列からなる文書処理記

述を生成し、該文書処理記述を解釈して構造化文書を合成処理する文書処理手段と、
前記文書処理手段で処理して得られた構造化文書又は文書部品を、所定フォーマットのファイルとしてネットワーク経由で出力する出力手段とを具備することを特徴とする構造化文書処理システム。
【請求項８】処理起動記述は構造化文書処理サーバのサーバ名を含む形式で前記ネットワーク上の分散ファイル名を規定することができ、
前記ネットワーク上には少なくとも構造化文書を処理する第１及び第２の構造化文書処理サーバが存在し、
第１の構造化文書処理サーバに入力される第１の処理起動記述では、文書処理の対象となる第１の原料文書及び／又は第１の雛型文書のファイル名が第２の構造化文書処理サーバのサーバ名を含む第２の処理起動記述の形式で記述され、
第１の構造化文書処理サーバは、第１の処理起動記述を入力したことに応答して、第１の原料文書及び／又は第１の雛型文書のファイル名として記述された第２の処理起動記述を抜き出して第２の構造化文書処理サーバに前記ネットワーク経由で送信するとともに、第２の構造化文書処理サーバが第２の処理起動記述を起動して出力する構造化文書又は文書部品を含むファイルをネットワーク経由で受信して第１の原料文書及び／又は第１の雛型文書として使用する、ことを特徴とする請求項７に記載の構造化文書処理システム。
【請求項９】第２の処理起動記述を入力する第２の構造化文書処理サーバは、前記ネットワーク経由の通信を必要としない第１の構造化文書処理サーバと同一のコンピュータ・システム上で構成され、
第１の構造化文書処理サーバは、構造化文書又は文書部品を含むファイルに替えて、第２の構造化文書処理サーバによる処理結果である構造化文書又は文書部品を構文解析木として入力する切り替え手段を具備する、ことを特徴とする請求項８に記載の構造化文書処理システム。
【請求項１０】前記ファイル・サーバから入力した原料文書又は雛型文書の構文解析木をファイル名又は処理起動記述と対応付けて保持する保持手段と、
ファイル名に対応する構造化文書ファイルを前記ファイル・サーバから入力する代わりに、該当する構文解析木を前記保持手段から入力する入力手段と、をさらに具備することを特徴とする請求項７に記載の構造化文書処理システム。
【請求項１１】前記構造化文書処理システムは、文書部品の抽出を指定した抽出命令が付与された第１の構造化文書のファイル名と反復複写命令又は属性置換命令を付与した第２の構造化文書のファイル名を含む処理起動記述を入力し、
前記属性置換手段は、前記処理起動記述の一部を、文書部品にあらかじめ設定されている属性文字列と、置き換えて得られる文字列を、該文書部品の属性文字列として設定することを特徴とする請求項２に記載の構造化文書処理システム。
【請求項１２】構造的に表現され１以上の文書部品を含んだ構造化文書を処理するための構造化文書処理システムであって、
構造化文書の構造を解析して構文解析木を生成する解析手段と、
前記解析手段により生成された構文解析木を基に構造化文書に付与された命令と文書部品を分離して命令を取り出すとともに、命令の文法的なエラーを検出してエラー情報を出力する命令分離手段と、
エラー情報を入力してエラー通知を行なう文書を合成するエラー通知文書合成手段と、
構造化文書から取り出された各命令を併合・整列して構造化文書を処理するための命令列からなる文書処理記述を生成するとともに、エラー通知文書へのアクセス情報を生成する処理起動記述合成手段と、
処理起動記述を解釈して、エラー通知文書を取り出す処理起動記述解析手段と、
エラー通知文書を保持する保持手段と、を具備することを特徴とする構造化文書処理システム。
【請求項１３】構造的に表現され１以上の文書部品を含んだ構造化文書を処理するための構造化文書処理方法であって、
命令が付与された構造化文書を解析して構文解析木を生成するステップと、
構文解析木を探索して、文書部品と命令とを分離して命令を取り出すステップと、
構造化文書から取り出された各命令を併合・整列して、構造化文書を処理するための命令列からなる文書処理記述を生成するステップ、
文書処理記述を解釈して、構造化文書を合成処理するステップと、を具備することを特徴とする構造化文書処理方法。
【請求項１４】構造的に表現され１以上の文書部品を含んだ構造化文書を処理するための構造化文書処理方法であって、（ａ）文書部品の抽出を指定する抽出命令が付与された第１の構造化文書を解析して構文解析木を生成するステップと、（ｂ）文書部品の所定回数の複写を指定する反復複写命令又は文書部品の挿入若しくは置換を指定する挿入置換命令が付与された第２の構造化文書を解析して構文解析木を生成するステップと、（ｃ）構文解析木を探索して、文書部品と命令とを分離して命令を取り出すステップと、（ｄ）抽出命令が付与された第１の構造化文書から取り出された抽出命令と、反復複写命令及び／又は挿入置換命令が付与された第２の構造化文書から取り出された反復複写命令及び／又は挿入置換命令とを併合・整列して、該第１及び第２の構造化文書を処理するための命令列からなる文書処理記述を生成する

ステップ、（ｅ）文書処理記述を解釈して、抽出命令の対象である第１の文書部品を第１の構造化文書から取り出すステップと、（ｆ）文書処理記述を解釈して、反復複写命令の対象である文書部品と該文書部品付与された命令を所定回数だけ反復複写するステップと、（ｇ）文書処理記述を解釈して、第１の文書部品を、挿入置換命令の対象である第２の文書部品の前又は後に挿入するか又は該第２の文書部品と置換するステップと、（ｈ）前記ステップ（ｅ）乃至（ｇ）の結果得られた構文解析木を出力するステップと、を具備することを特徴とする構造化文書処理方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、文書を章、節、段落、図表、あるいは表題や章題、概要などの複数の素片に分解し各素片をノードとするツリー構造やグラフ構造などのように構造的に表現して取り扱う構造化文書の処理技術に係り、特に、複数の構造化文書を基にして新たに文書を合成する構造化文書の処理技術に関する。

【０００２】更に詳しくは、本発明は、複数の構造文書から特定の条件を満たす文書部分（「文書部品」）を取り出すとともに文書部品を別の文書中に挿入又は置換して合成を行なう構造化文書の処理技術に係り、構造化文書から文書部品を抽出したり、各文書部品を雛型文書中に挿入又は置換したりする手続きを記述したスクリプトを用いることなしに文書の合成処理を行う構造化文書の処理技術に関する。

【０００３】

【従来の技術】一般に、文書がテキスト文字列のみで構成されることは稀であり、章や節、段落という区切りを持ったり、図表などのコンテンツが挿入されていたり、あるいは表題や章題、概要などの素片を含むことが多い。

【０００４】このため、単一の文書を章、節、段落、図表、あるいは表題や章題、概要などの複数の素片に分解し、各素片をノードとするツリー構造やグラフ構造などのように構造的に表現して取り扱う文書処理技術が開発されてきた。このように構造化された文書は一般に「構造化文書」と呼ばれ、計算機システムを用いてさまざまの処理を行うことが可能である。

【０００５】構造化文書では、各ノードとリンクで表現される親子関係は、文書の論理的な構造を表現する。例えば「章題」「図」「章」などの属性が付与されたノードに対して、印刷のため各ノードの単位でレイアウト処理を行なったり、版下作成の処理や、ノードの属性を手がかりにして抄録集や目次を自動的に作成する処理などが可能になる。

【０００６】現在、構造化文書を記述するフォーマットとしては、ＳＧＭＬ（Ｓｔａｎｄａｒｄ Ｇｅｎｅｒａｌｉｚｅｄ Ｍａｒｋｕｐ Ｌａｎｇｕａｇｅ）やＸＭＬ（ｅＸｔｅｎｓｉｂｌｅ Ｍａｒｋｕｐ Ｌａｎｇｕａｇｅ）ＨＴＭＬ（Ｈｙｐｅｒ Ｔｅｘｔ Ｍａｒｋｕｐ Ｌａｎｇｕａｇｅ）などの記述言語がよく知られている。例えば、ＨＴＭＬは、表（ＴＡＢＬＥ）や箇条書き（ＵＬ）を指示する記法がある。

【０００７】プレーンなテキスト・ファイルに構造を指示する方法の一つとして、マークアップと呼ばれる方法がある。マークアップは、あらかじめ取り決められた規則に基づき、特定の論理的な構造の開始を表すタグ記号と終了を表すタグ記号で文書の一部分を挟むことで構造を定める。例えばＨＴＭＬでは、箇条書きの開始タグを＜ＵＬ＞とし、終了タグを＜／ＵＬ＞として、箇条を立てる項目の開始タグを＜ＬＩ＞終了タグを＜／ＬＩ＞とすると、次のように表記される。

【０００８】

【数１】＜ＵＬ＞
＜ＬＩ＞箇条書き項目１＜／ＬＩ＞
＜ＬＩ＞箇条書き項目２＜／ＬＩ＞
＜ＬＩ＞箇条書き項目Ｎ＜／ＬＩ＞
＜／ＵＬ＞

【０００９】このような構造化文書に対して、文書の構造を利用した検索を行なって、複数の文書から特定の条件を満たす文書部分（以下では、「文書部品」と呼ぶ）を取り出し、取り出した文書部品を別の文書中に挿入して合成を行なう方式が提案されている。例えば、本出願人に既に譲渡されている特開平６－５２１６１号公報に「文書処理方法及び文書処理装置」として開示される文書処理方式では、ツリーまたはグラフで表現された構造化文書に対して、ノードとして表される文書部品（同公報中では「構成子」と呼ばれている）のタイプを指定して検索するｓｅｌｅｃｔ－ｔｙｐｅ命令によって、所定の属性を持つ文書部品を取り出して、第２の文書に挿入することが記述されている。例えばｓｅｌｅｃｔ－ｔｙｐｅ命令の引数として、’Ｆｉｇｕｒｅ’や’Ｓｅｇｍｅｎｔ’を指定することで、それぞれ図と節を検索して取り出す例が開示されている。また、文書部品の挿入に関しては、挿入先の文書の特定のノードを指定し、指定したノードの末子に挿入する例が記述されている。また、ノードの親子関係を判定して、これによって特定の構造パターンを有するノードを検索することができる。

【００１０】富士ゼロックス株式会社の製品”Ａｋａｎｅ”は、ウィンドウ・システム上で動作する構造化文書エディタを中心とした文書処理アプリケーション・ソフトウェアである。Ａｋａｎｅの周辺ツールとして、文書処理コマンド・セットが提供されている。「Ａｋａｎｅ文書操作コマンドセットプログラマーズガイド」第３章応用事例（ｐ．２－９５～２－９６）には、文書の論理構造をパターンで指定して、特定の条件を満たす構造を有するノードを文書部品として取り出し、一つの文書として合成する例が記述されている。

【0011】このように、あらかじめ検索式を指定したプログラムと、検索結果として取り出された文書部品を処理するプログラムをパイプラインで結合し、入力とする原料文書から、特定の条件を満たす文書部品を取り出して新たな構造化文書や文書部品を合成することができる。

【0012】XMLが構造化文書を記述可能な言語であることは既に述べた通りである。「XML開発事例」（株式会社ASCII ISBN4-7561-3112-3）には、XMLで記述された構造化文書を入力し、処理を行なうXSL（eXtensible Stylesheet Language）言語とそのプロセッサによる構造化文書の扱いが開示されている。XSLの構文は、例えば次のような構造をとる。

【0013】

【数2】<rule>
[pattern]
[action]
</rule>

【0014】ここで、[pattern]は、処理の対象とする文書部品の検索式を記述する。また、[action]は、検索された文書部品に対する処理を記述する。検索式の例を以下に示す。

【0015】

【数3】<rule>
<target-element type = "section"/>
<element type = "figure"/>
[action]
</rule>

【0016】<target-element type="section"/>は、検索する文書部品のノードの型が"section"であり、次の<element type = "figure">は、文書部品のノードの型が"figure"である子ノードを含むように限定することを示す検索式である。

【0017】また、次に示す例は、ノードの型が"employee"であって、かつ、親ノードの型が"person"であるような文書部品を検索する式である。

【0018】

【数4】<rule>
<element type = "employee"/>
<target-element type = "person"/>
[action]
</rule>

【0019】このように、検索式と、検索結果として取り出された文書部品を処理するアクションとをあらかじめ記述したスクリプトを解釈実行することで、入力とする原料文書から、特定の条件を満たす文書部品を取り出して新たな構造化文書や文書部品を合成することができる。

【0020】また、本出願人に既に譲渡されている特開平7-56920号公報において「構造化文書処理装置」として開示される文書処理装置は、構造化文書から複数の文書部品を抽出して列にして出力する部分構造列抽出部と、文書部品の列を入力して処理を行う処理実行部を備えている。かかる構成の文書処理装置によれば、抽出指示部と処理指示部を別々に管理することで、原料とする文書の構造の変更に対して、文書処理の変更を抽出指示部だけにとどめることができる。例えば、上記した各従来技術に対してシステムや文書の改変と維持管理が簡単になる。

【0021】

【発明が解決しようとする課題】上述した従来技術は、要するに図26又は図27のようなシステム構成となる。例えば、図26に示す構造文書処理システムの場合、まず、原料文書と雛型文書（テンプレート）が抽出／合成プログラムに投入される。抽出／合成プログラムは、原料文書からの文書部品の抽出や雛型文書への文書部品の挿入・置換などの手続きをスクリプト形式で記述した「抽出／合成スクリプト」に従って処理を実行し、合成文書を生成する。

【0022】また、図27に示す構造文書処理システムの場合、まず、抽出プログラムが抽出スクリプトで記述された手続きに従い原料プログラムから文書部品の抽出処理を行う。抽出された各文書部品は、雛型文書とともに合成プログラムに投入される。そして、合成プログラムは、合成スクリプトで記述された手続きに従って各文書部品の雛型文書への挿入・置換処理を実行して、合成文書を生成する。

【0023】これら上記した技術はいずれも、原料文書から文書部品を抽出するための手続きを記述したスクリプト（抽出スクリプト）や、結果として出力する文書の基になる雛形文書に対して、原料文書から抽出した文書部品を挿入または置換するためのスクリプト（合成スクリプト）を使用するものである。言い換えれば、これら従来技術は、元の文書とは別にこれらスクリプトを管理する必要があり、以下に示す問題を包含する。

【0024】1. 原料から文書部品を取り出すため、文書部品の構造やパターンを特定する検索式をスクリプト中に記述する必要がある。したがって、原料文書の構造を変更すると、スクリプト中の検索式もこれに合わせて変更しなくてはならない。

【0025】2. 構造が異なる複数の原料文書を混在させて処理するには、異なる構造ごとにスクリプトを用意しなくてはならない。

【0026】3. 文書部品の構造やパターンを特定する

検索式や、取り出した文書部品を処理する手順を手続きとして記述することが難しい。一般には、「条件Aを満たす文書部品を検索して、得られた結果に処理Bを行なう」という手続き的な指示をスクリプトとして記述する必要がある。また、文書部品の数が原料文書によって変わる場合に、文書部品の数を数え、繰り返しを行なうｆｏｒ文やｒｅｐｅａｔ文などの繰り返し命令と、所望の処理を実際に行なう命令（例えば挿入命令）とを組み合わせて、繰り返し処理（例えば挿入や置換）を行なう指示をスクリプトとして記述する必要がある。このような記述やスクリプトの作成には、プログラミングの相当程度の知識が必要であり、一般ユーザが広く行なうことは難しく、あまり普及しないと思料される。

【００２７】４．　文書処理の中間結果（例えば検索処理によって抽出された文書部品）を簡単に再利用する機構について言及していない。例えば、上述したＡｋａｎｅの場合には、中間結果をファイルに保存するスクリプトを明示的に記述する必要がある。

【００２８】第１の問題点に関しては、原料文書の構造を変更した際に、これを処理するスクリプトを網羅的に探したり、スクリプトの変更を忘れると文書処理が動作不良を起こすなどの不具合が発生する。

【００２９】また、第２の問題点に関しては、構造が相違する入力文書ごとにそれを処理する専用のスクリプトの開発とその維持管理に手間や労力が掛かるという不具合がある。また、利用者にとっては、適切なスクリプトを選んで利用する必要がある。スクリプトの選択を誤ると、文書処理が正しく動作しないなどの不具合が発生する。

【００３０】また、第３の問題点に関しては、それぞれの目的に適合した構造化文書処理のアプリケーションを作成することができる文書処理システムを、利用者自らが実装することが極めて困難となる。

【００３１】また、第４の問題点に関しては、効率の良いアプリケーションの開発に手間がかかるという不具合がある。

【００３２】本発明は上述のような技術的課題を勘案したものであり、その目的は、文書を章、節、段落、図表、あるいは表題や章題、概要などの複数の素片に分解し各素片をノードとするツリー構造やグラフ構造などのように構造的に表現して取り扱うことができる、優れた構造化文書処理システム及び構造化文書処理方法を提供することにある。

【００３３】本発明の更なる目的は、複数の構造化文書を基にして新たに文書を合成することができる、優れた構造化文書処理システム及び構造化文書処理方法を提供することにある。

【００３４】本発明の更なる目的は、複数の構造化文書から特定の条件を満たす文書部分（「文書部品」）を取り出すとともに文書部品を別の文書中に挿入又は置換して合成を行なうことができる、優れた構造化文書処理システム及び構造化文書処理方法を提供することにある。

【００３５】本発明の更なる目的は、構造化文書から文書部品を抽出したり、各文書部品を雛型文書中に挿入又は置換したりする手続きを記述したスクリプトを用いることなしに構造化文書の合成処理を行うことができる、優れた構造化文書処理システム及び構造化文書処理方法を提供することにある。

【００３６】

【課題を解決するための手段】本発明は、上記課題を参酌しなされたものであり、その第１の側面は、構造的に表現され１以上の文書部品を含んだ構造化文書を処理するための構造化文書処理システムであって、文書部品の抽出を指定する抽出命令又は文書部品の所定回数の複写を指定する反復複写命令又は文書部品の挿入若しくは置換を指定する挿入置換命令を該当する文書部品に付与する命令付与手段と、構造化文書の構造を解析して構文解析木を生成する解析手段と、前記解析手段により生成された構文解析木を基に構造化文書に付与された命令と文書部品を分離して命令を取り出す命令分離手段と、抽出命令が付与された第１の構造化文書から前記命令分離手段によって取り出された抽出命令と、反復複写命令及び／又は挿入置換命令が付与された第２の構造化文書から前記命令分離手段によって取り出された反復複写命令及び／又は挿入置換命令とを併合・整列して、該第１及び第２の構造化文書を処理するための命令列からなる文書処理記述を生成する文書処理記述生成手段と、抽出命令の対象である第１の文書部品を第１の構造化文書から取り出す抽出手段と、反復複写命令の対象である文書部品と該文書部品付与された命令を所定回数だけ反復複写する反復複写手段と、第１の文書部品を、挿入置換命令の対象である第２の文書部品の前又は後に挿入するか又は該第２の文書部品と置換する挿入置換手段と、文書処理記述を順次解釈して、前記抽出手段、前記反復複写手段、及び／又は前記挿入置換手段を用いて文書部品を合成するインタプリタと、を具備することを特徴とする構造化文書処理システムである。

【００３７】本発明の第１の側面に係る構造化文書処理システムは、命令付与手段によって文書部品の抽出を指定する抽出命令や、文書部品の所定回数の複写を指定する反復複写命令、文書部品の挿入若しくは置換を指定する挿入置換命令を、構造化文書中の該当する文書部品に付与することができる。

【００３８】また、この構造化文書処理システムに対して、抽出命令が付与された第１の構造化文書と、反復複写命令及び／又は挿入置換命令が付与された第２の構造化文書の各々を投入すると、解析手段によって各構造化文書毎に文書解析木が構成され、命令分離手段によって各構造化文書に付与された命令列を取り出される。さらに、取り出された複数の命令列の併合・整列、及び変換

が行なわれ、文書処理記述が文書処理記述合成手段によって合成される。

【００３９】インタプリタは、文書処理記述を順に走査して、文書処理記述中に含まれる各命令に従って抽出手段によって第１の文書部品を第１の文書から抽出し、反復複写手段によって第２の構造化文書の文書部品を第１の文書部品の数に依存して定まる回数だけ反復複写し、また、挿入置換手段によって第２の構造化文書に第１の文書部品を挿入又は置換することで、構造化文書の合成を行なうことができる。

【００４０】本発明の第１の側面に係る構造化文書処理システムによれば、以下に示す作用効果を奏することができる。すなわち、

【００４１】１．　原料文書の構造を変更した際に、これを処理するための別に管理されるスクリプトを用意しなくてよい。

２．　構造が異なる複数の原料文書を混在させて文書部品を取り出すには、各文書毎に文書部品に抽出命令を付与することで行なうので、文書毎の抽出の処理を個別に指定しなくてよい。同様に、文書部品を挿入又は置換する処理の指定も不要である。また、ユーザは、文書の構造が変わる度に適切なスクリプトを指定する必要がない。

３．文書部品は、所望の命令を元の文書中に直接的に付与することと、取り出した文書部品と挿入置換される文書部品の条件の指定（例えばラベル文字列が一致するなど）によって作成することができる。また、各ユーザがそれぞれの目的に応じて雛型文書をデザイン（作成）したり、所望の処理を行なう命令が組み込まれた雛型文書を選択して、ユーザ自らが原料文書と組み合わせることで文書処理アプリケーションを構成することができる。このように、手続き的でない、言い換えれば宣言的な方法で文書処理を構成することができるので、プログラミングの知識を充分に持たないユーザが広く文書処理を行なうことができる。すなわち、各ユーザがそれぞれの目的に合わせた構造化文書処理のアプリケーションを容易に作成することができる構造化文書処理システムが実現できる。

【００４２】本発明の第１の側面に係る構造化文書処理システムにおいて、前記抽出命令は文書部品の属性の取り出しを指定する属性抽出命令であり、また、前記挿入置換命令は文書部品の属性の置換を指定する属性置換命令であってもよい。このような場合、前記命令分離手段は構造化文書から属性抽出命令と属性挿入置換命令を取り出し、前記挿入置換手段は第１の文書部品の属性を属性置換命令の対象である第２の文書部品の属性と置換する属性置換手段であるとともに、前記インタプリタは前記文書処理記述合成手段が合成した文書処理記述を解釈して属性抽出命令で指定された文書部品の所定のノードの属性を抽出するようにしてもよい。この結果、該抽出された属性を属性置換命令で指定された文書部品の所定のノードに設定することができる。

【００４３】この場合、前記属性置換手段は、属性置換命令に対して、文書部品にあらかじめ設定されている属性値文字列とシステムの状態から得られる文字列を組み合わせて合成される文字列に置き換える属性置換手段であってもよい。

【００４４】あるいは、前記抽出命令はパス名付き抽出命令であり、また、前記反復複写命令及び挿入置換命令はそれぞれパターン式付き反復複写命令及びパターン式付き挿入置換命令であってもよい。このような場合、前記命令分離手段は、構造化文書から、抽出命令とパス名を取り出すか、又は、反復複写命令と挿入置換命令とパターン式を取り出し、前記反復複写手段は、抽出した文書部品に付与されたパス名とパターン式とのパターン・マッチングを行い、マッチするパス名を持つ文書部品の個数だけ反復複写を実行し、また、前記挿入置換手段は、抽出した文書部品に付与されたパス名とパターン式とのパターン・マッチングを行い、マッチするパス名を持つ文書部品を挿入又は置換するようにすればよい。

【００４５】また、前記挿入置換手段は、前記抽出手段によって取り出された文書部品に付与された抽出命令を挿入又は置換するようにしてもよい。この結果、構造化文書処理システムを複数段接続してパイプラインを構成することができる。この機能によって、より柔軟で複雑な構造化文書処理を実現することが可能となる。

【００４６】このような場合、前記挿入置換手段は、抽出命令を挿入又は置換する際に、パス名付き抽出命令のパス名を変更してから挿入置換するようにしてもよい。この結果、最初の構造化文書処理システムで出力された原料文書に含まれている文書部品は、２段目以降の構造化文書処理ではパス名を調べることで識別することができる。

【００４７】また、本発明の第２の側面は、ネットワーク接続された２以上のコンピュータで構成される分散ネットワーク・システム上におけるコンピュータ通信による協調的な処理によって実現される、構造的に表現され１以上の文書部品を含んだ構造化文書処理システムであって、構造化文書を所定フォーマットのファイルとして格納するとともに、ファイル名を受信したことに応答して該当するファイルを前記ネットワーク経由で送信するファイル・サーバと、ファイルに対して文書処理を行う構造化文書処理サーバとを少なくとも含み、前記構造化文書処理サーバは、文書部品の抽出を指定した抽出命令が付与された第１の構造化文書のファイル名と反復複写命令又は挿入置換命令を付与した第２の構造化文書のファイル名を含む処理起動記述を入力し解析して、該処理起動記述に含まれるファイル名を前記ファイル・サーバに前記ネットワーク経由で送信し、ファイル名に該当する各ファイルを前記ファイル・サーバからネットワーク

経由で入力する入力手段と、第１の構造化文書及び第２の構造化文書を解析して構文解析木を生成し、構文解析木を探索して文書部品と命令とを分離して命令を取り出し、各命令を併合・整列して構造化文書を処理するための命令列からなる文書処理記述を生成し、該文書処理記述を解釈して構造化文書を合成処理する文書処理手段と、前記文書処理手段で処理して得られた構造化文書又は文書部品を、所定フォーマットのファイルとしてネットワーク経由で出力する出力手段とを具備することを特徴とする構造化文書処理システムである。

【００４８】ここで言うネットワークとは、例えば、インターネットのように複数のコンピュータがＴＣＰ／ＩＰ（Ｔｒａｎｓｍｉｓｓｉｏｎ Ｃｏｎｔｒｏｌ Ｐｒｏｔｏｃｏｌ／Ｉｎｔｅｒｎｅｔ Ｐｒｏｔｏｃｏｌ）接続して構成される分散ネットワークである。このような場合、構造化文書は、例えばＨＴＭＬ（ＨｙｐｅｒＴｅｘｔ Ｍａｒｋｕｐ Ｌａｎｇｕａｇｅ）のようなマークアップ言語によって記述される。また、ファイル・サーバは、ＨＴＴＰ（Ｈｙｐｅｒ ＴｅｘｔＴｒａｎｓｆｅｒ Ｐｒｏｔｏｃｏｌ）プロトコルに従ってＨＴＭＬファイルを提供するＨＴＴＰサーバとして構成される。また、ファイル名は、ＵＲＬ（Ｕｎｉｆｏｒｍ Ｒｅｓｏｕｒｃｅ Ｌｏｃａｔｏｒ）のような資源アクセス識別子の形式で表記される。

【００４９】また、構造化文書処理サーバは、ＨＴＴＰリクエストの形式で処理起動記述を入力することができる。この処理起動記述は、ＨＴＴＰリクエストのＵＲＬ中に別のＵＲＬを埋め込んだ記述形式の「ヴァーチャルＵＲＬ」として構成される。より具体的には、第１の構造化文書及び第２の構造化文書それぞれのファイル名、すなわちＵＲＬを含んだヴァーチャルＵＲＬである。したがって、構造化文書処理サーバは、このようなヴァーチャルＵＲＬを基にして、該当するファイル・サーバすなわちＨＴＴＰサーバから第１及び第２の構造化文書をネットワーク経由で取得することができる。

【００５０】さらに、構造化文書処理サーバは、第１の構造化文書及び第２の構造化文書を解析してそれぞれの構文解析木を生成し、構文解析木を探索して文書部品と命令とを分離して命令を取り出し、各命令を併合・整列して構造化文書を処理するための命令列からなる文書処理記述を生成し、該文書処理記述を解釈して、構造化文書を合成処理することができる。

【００５１】このようにして得られた構造化文書又は文書部品は、ＨＴＭＬフォーマットのファイルとして、すなわちＨＴＴＰレスポンスとしてネットワーク経由で出力することができる。

【００５２】本発明の第２の側面に係る構造化文書処理システムでは、処理起動記述は構造化文書処理サーバのサーバ名を含む形式で前記ネットワーク上の分散ファイル名を規定することを許容してもよい。

【００５３】例えば、前記ネットワーク上に構造化文書を処理する第１及び第２の構造化文書処理サーバが存在する場合、第１の構造化文書処理サーバに入力される第１の処理起動記述において、文書処理の対象となる第１の原料文書及び／又は第１の雛型文書のファイル名を第２の構造化文書処理サーバのサーバ名を含む第２の処理起動記述の形式で記述することができる。

【００５４】第１の構造化文書処理サーバは、第１の処理起動記述を入力したことに応答して、第１の原料文書及び／又は第１の雛型文書のファイル名として記述された第２の処理起動記述を抜き出して、これを第２の構造化文書処理サーバに前記ネットワーク経由で送信すればよい。第２の構造化文書処理サーバは、第２の処理起動記述を解釈して処理を起動し、処理結果としての構造化文書又は文書部品を含むファイルをネットワーク経由で返信する。第１の構造化文書処理サーバは、該ファイルを第２の構造化文書処理サーバから受信して、第１の原料文書及び／又は第１の雛型文書として使用することができる。

【００５５】また、第２の処理起動記述を入力する第２の構造化文書処理サーバは、前記ネットワーク経由の通信を必要としない第１の構造化文書処理サーバと同一のコンピュータ・システム上で構成されてもよい。この場合、第１の構造化文書処理サーバは、構造化文書又は文書部品を含むファイルに替えて、第２の構造化文書処理サーバによる処理結果である構造化文書又は文書部品を構文解析木として入力する切り替え手段を備えていてもよい。この結果、解析手段による重複した構文解析処理を省略することができ、処理速度が向上するとともに計算機負荷や通信負荷が軽減される。

【００５６】また、本発明の第２の側面に係る構造化文書処理システムは、さらに、前記ファイル・サーバから入力した原料文書又は雛型文書の構文解析木をファイル名又は処理起動記述と対応付けて保持する保持手段と、ファイル名に対応する構造化文書ファイルを前記ファイル・サーバから入力する代わりに該当する構文解析木を前記保持手段から入力する入力手段とを備えていてもよい。この結果、保持手段にキャッシュされた構文解析木を有効に活用して、処理速度を向上させることができる。

＃ｄ

また、本発明の第１の側面に係る構造化文書処理システムも、文書部品の抽出を指定した抽出命令が付与された第１の構造化文書のファイル名と反復複写命令又は属性置換命令を付与した第２の構造化文書のファイル名を含む処理起動記述を入力として用いることができる。この場合、属性置換手段は、処理起動記述の一部と、文書部品にあらかじめ設定されている属性文字列と、置き換えてえられる文字列を、該文書部品の属性文字列として設定するようにしてもよい。

【００５７】また、本発明の第３の側面は、構造的に表現され１以上の文書部品を含んだ構造化文書を処理するための構造化文書処理システムであって、構造化文書の構造を解析して構文解析木を生成する解析手段と、前記解析手段により生成された構文解析木を基に構造化文書に付与された命令と文書部品を分離して命令を取り出すとともに、命令の文法的なエラーを検出してエラー情報を出力する命令分離手段と、エラー情報を入力してエラー通知を行なう文書を合成するエラー通知文書合成手段と、エラー通知文書へのアクセス情報を生成する処理起動記述合成手段と、処理起動記述を解釈して、エラー通知文書を取り出す処理起動記述解析手段と、エラー通知文書を保持する保持手段と、を具備することを特徴とする構造化文書処理システムである。

【００５８】ここで言うエラー情報は、エラーの種類やエラーの検出された文書中の位置（行数）など、通常のエラー処理でよく扱われるものでよい。

【００５９】エラー通知文書合成手段は、エラー情報を入力して、エラー通知を行なう文書を合成する。エラー情報は例えばエラー番号とし、エラー文書はエラー番号をファイル名とするファイルとする。エラー通知文書合成手段は、命令解析手段から入力されたエラー番号をファイル名とみなして、ファイルからエラー通知文書の１つを入力する。入力された文書は、解析手段を用いて構文解析木が作られる。構文解析木は保持手段に保持されるが、必ずしも構文解析木を作る必要はなく、文字列情報のまま保持手段に保持してもよい。

【００６０】処理起動記述合成手段が生成するエラー通知文書へのアクセス情報は、例えば、エラー通知ページにアクセスするためのＵＲＬとして記述される。エラー通知文書は、処理起動記述合成手段で合成されたＵＲＬをキー情報として、保持手段に格納される。

【００６１】本発明の第３の側面によれば、文書処理の途中で発生したエラーを、文書処理の合成結果とは別にして利用者に通知することができる。したがって、高度で柔軟な文書処理アプリケーションを構成することができる。さらに、エラーが発生した時点でエラー通知文書を生成して、文書処理の結果を一時的に格納する保持手段に（エラー通知文書のリクエストに先行して）格納し、後のエラー通知文書の要求に対して出力するように動作することができる。このため、エラーを発生したプロセスとエラーの種類やエラーに関する情報を対応付けて保持するような機構を用意する必要がなくなる。

【００６２】また、本発明の第４の側面は、構造的に表現され１以上の文書部品を含んだ構造化文書を処理するための構造化文書処理方法であって、命令が付与された構造化文書を解析して構文解析木を生成するステップと、構文解析木を探索して、文書部品と命令とを分離して命令を取り出すステップと、構造化文書から取り出された各命令を併合・整列して、構造化文書を処理するための命令列からなる文書処理記述を生成するステップ、文書処理記述を解釈して、構造化文書を合成処理するステップと、を具備することを特徴とする構造化文書処理方法である。

【００６３】また、本発明の第５の側面は、構造的に表現され１以上の文書部品を含んだ構造化文書を処理するための構造化文書処理方法であって、（ａ）文書部品の抽出を指定する抽出命令が付与された第１の構造化文書を解析して構文解析木を生成するステップと、（ｂ）文書部品の所定回数の複写を指定する反復複写命令又は文書部品の挿入若しくは置換を指定する挿入置換命令が付与された第２の構造化文書を解析して構文解析木を生成するステップと、（ｃ）構文解析木を探索して、文書部品と命令とを分離して命令を取り出すステップと、（ｄ）抽出命令が付与された第１の構造化文書から取り出された抽出命令と、反復複写命令及び／又は挿入置換命令が付与された第２の構造化文書から取り出された反復複写命令及び／又は挿入置換命令とを併合・整列して、該第１及び第２の構造化文書を処理するための命令列からなる文書処理記述を生成するステップ、（ｅ）文書処理記述を解釈して、抽出命令の対象である第１の文書部品を第１の構造化文書から取り出すステップと、（ｆ）文書処理記述を解釈して、反復複写命令の対象である文書部品と該文書部品付与された命令を所定回数だけ反復複写するステップと、（ｇ）文書処理記述を解釈して、第１の文書部品を、挿入置換命令の対象である第２の文書部品の前又は後に挿入するか又は該第２の文書部品と置換するステップと、（ｈ）前記ステップ（ｅ）乃至（ｇ）の結果得られた構文解析木を出力するステップと、を具備することを特徴とする構造化文書処理方法である。

【００６４】

【作用】従来の構造化文書処理方式では、原料文書や雛形文書と別に文書を処理するためのスクリプトを作成し、これを管理することが問題の原因であった。

【００６５】これに対し、本発明では、スクリプトのように手続きの体裁を備えた命令を用いずに構造化文書の処理を行うものである。

【００６６】図１には、本発明に係る構造化文書処理システムの基本概念を図解している。同図に示すように、この構造化文書処理システムは、命令付与手段と、解析手段と、命令分離手段と、文書処理記述合成手段と、インタプリタと、抽出手段と、反復複写手段と、挿入置換手段とで構成される。これら各手段は、例えば、計算機システム上の処理プロセスとして実現することができる。

【００６７】命令付与手段は、原料文書としての構造化文書内に各命令を埋め込む。ここで言う命令には、構造化文書の文書部品の取り出しを指定する抽出命令と、文書部品を所定の回数の反復複写の指定をする反復複写命

令と、文書部品の挿入又は置換を指定する挿入置換命令が含まれる。

【００６８】これら各命令はそれ自体では、スクリプトのように処理を記述した「手続き」としての体裁を整えていない。例えば、構造化文書がＨＴＭＬやＸＭＬなどのマークアップ言語で記述されている場合には、いわゆるコメント文の形式で元の文書内に埋め込まれる。

【００６９】解析手段は、各命令が埋め込まれた第１の構造化文書と第２の構造化文書とを受け取って、これらを解析する。

【００７０】命令分離手段は、入力された第１及び第２の構造化文書を文書部品と命令に分離して、各命令を取り出す。

【００７１】文書処理記述合成手段は、取り出した各命令を併合し、所定の順に並べて文書処理記述を合成する。

【００７２】インタプリタは、一般に、前記文書処理記述に含まれる命令列を順に解釈実行する。本発明においては、インタプリタは、抽出手段に文書部品を指定して、原料文書から文書部品を取り出させる。また、インタプリタは、反復複写手段に文書部品を指定して、文書部品や各命令を反復複写せしめる。さらに、挿入置換手段に２つの文書部品を指定して、文書部品の挿入又は置換を行なわしめる。この結果、文書部品同士を合成するという構造化文書の処理が実現する。

【００７３】本発明に係る構造化文書処理システムに入力されたそれぞれの構造化文書中には、文書部品を取り出す抽出命令と、反復複写と挿入置換命令が付与されている。したがって、文書部品の取り出しの指定と、反復複写の指定と、文書部品を挿入または置換する文書部品（個所）の指定を行ない、入力した複数の構造化文書から取り出した命令を動的に合成して文書処理記述を作成するようにした結果、文書処理記述スクリプトが不要となる。したがって、元の文書と別にスクリプトを管理するという手間が省略される。

【００７４】本発明によれば、文書処理記述は、入力文書の構造に合わせた抽出命令（列）と、反復複写命令（列）や挿入置換命令（列）とから合成される。本発明に係る構造化文書処理システムに構造化文書が入力されると、命令（列）が文書から分離され併合され、文書を処理する前に文書処理記述が動的に合成される。したがって、合成された文書処理記述は、複数の入力文書の処理に適合した処理記述となっている。

【００７５】本発明によれば、文書処理の度に入力文書に合わせた処理記述が動的に構成される。したがって、従来は必要であった文書処理スクリプトが不要となり、必然的に、文書と別にスクリプトを管理するという労力からも解放される。

【００７６】本発明によれば、スクリプトを開発するためのプログラミングが不要になる。また、構造的に異なる文書が入力される毎に異なるスクリプトを用意する必要がない。利用者は、これらのスクリプトを入力文書に合わせて選択する必要がない。

【００７７】また、入力文書の構造の変更に伴うスクリプトの変更の必要がないので、変更を必要とするスクリプトを探したり、変更し忘れによる不具合が発生しない。したがって、構造化文書の利用者にとっては、それぞれの目的に合わせた構造化文書処理のアプリケーションを作成して、出力結果を所望の目的に合致した形に簡単にカスタマイズできるシステムの実現が容易になる。すなわち、効率のよいアプリケーションの開発が簡単になる。

【００７８】総じて、本発明によれば、構造化文書処理システムの開発や維持管理や利用が簡単である。

【００７９】本発明のさらに他の目的、特徴や利点は、後述する本発明の実施例や添付する図面に基づくより詳細な説明によって明らかになるであろう。

【００８０】

【発明の実施の形態】以下では、説明の便宜上、構造化文書（以下、単に「文書」とする）としてＨＴＭＬ（Ｈｙｐｅｒ　Ｔｅｘｔ　Ｍａｒｋｕｐ　Ｌａｎｇｕａｇｅ）言語で記述されたＨＴＭＬファイルであるとする。但し、ＸＭＬ（ｅＸｔｅｎｓｉｂｌｅＭａｒｋｕｐ　Ｌａｎｇｕａｇｅ）やその他の言語で記述された構造化文書に対して本発明を適用できることは言うまでもない。

【００８１】図２には、本発明の第１の実施例に係る構造化文書処理システムの構成を模式的に示している。以下、同図を参照しながら、各ブロックについて説明する。

【００８２】命令付与手段は、入力される文書内に、抽出命令、反復複写命令、挿入置換命令（以下では、これらをまとめて単に「各命令」とする）を埋め込む。各命令は、次のように記述される。すなわち、

【００８３】

【数５】・抽出命令

＜！－－＃ＯＵＴ　ＬＡＢＥＬ－－＞

※ＨＴＭＬで記述された文書部品

＜！－－＃／ＯＵＴ－－＞

【００８４】

【数６】・反復複写命令

＜！－－＃ＲＰ　ＬＡＢＥＬ－－＞

※ＨＴＭＬで記述された文書部品

＜！－－／＃ＲＰ－－＞

【００８５】

【数７】・挿入置換命令

＜！－－＃ＩＮ　ＬＡＢＥＬ－－＞

※ＨＴＭＬで記述された文書部品

＜！－－＃／ＩＮ－－＞

【００８６】この例では、’ＯＵＴ’が抽出命令を、’ＲＰ’が反復複写命令を、’ＩＮ’が挿入置換命令を、

それぞれ表すものとする。また、LABELは任意の文字列でよい。<!--#OUT LABEL-->を抽出命令開始タグ、<!--#/OUT-->を抽出命令終了タグと呼ぶ。同様に、<!--#RP LABEL-->及び<!--#/RP-->を反復複写命令開始タグ及び反復複写命令終了タグとし、<!--#IN LABEL-->及び<!--#/IN-->を挿入置換命令開始タグ及び挿入置換命令終了タグとする。挿入置換命令開始タグと挿入置換命令終了タグの間には、何も書かなくてもよい。その場合には、文書合成処理以前には正しいHTMLファイルにならない場合もある。

【0087】本実施例では、抽出命令を付与した構造化文書を原料文書とする。第1の原料文書のHTMLファイルを以下に示す。

【0088】

【数8】

```
<HTML>
<HEAD>
<TITLE> Sample1 </TITLE>
</HEAD>
<BODY>
<FONT SIZE="5"><!--#OUT グループ-->Aチーム<!--#/OUT-->10月度報告</FONT>
<H2>概要</H2>
<HR>
<!--#OUT サマリ-->
<UL>
<LI> 項目1の進捗60% 計画どおり
<LI> 項目2の進捗30% 遅れ: 3日
<LI> 項目3の進捗70% 予定外の問題発生、計画見直しが必要
</UL>
<!--#/OUT-->
<HR>
</BODY>
</HTML>
```

【0089】第1の原料文書を解析木の形式で表現すると図3のようになる。また、この第1の原料文書を標準的なHTMLブラウザに表示した結果は、図18に示した通りとなる。該原料文書に埋め込まれた抽出命令"OUT"は、このうち破線で囲まれた部分の取り出しを要求するものである。

【0090】また、第2の原料文書のHTMLファイルを以下に示す。

【0091】

【数9】

```
<HTML>
<HEAD>
<TITLE> Sample2 </TITLE>
</HEAD>
<BODY>
<H1>月度報告<!--#OUT グループ-->Bチーム<!--#/OUT--></H1>
<H2>概要</H2>
<HR>
<!--#OUT サマリ-->
  <TABLE BORDER="0" CELLSPACING="0" CELLPADDING="1">
    <TR><TH>項目</TH><TH>進捗</TH><TH ALIGN=LEFT">評価</TH></TR>
    <TR><TH>1</TH><TD>20%</TD><TD>やや計画おくれ(1日)</TD></TR>
    <TR><TH>2</TH><TD>40%</TD><TD>遅れなし</TD></TR>
    <TR><TH>3</TH><TD>60%</TD><TD>進み: 3日</TD></TR>
  </TABLE>
<!--#/OUT-->
<HR>
</BODY>
</HTML>
```

【0092】第2の原料文書を解析木の形式で表現すると図4のようになる。また、この第2の原料文書を標準的なHTMLブラウザに表示した結果は、図19に示した通りとなる。該原料文書に埋め込まれた抽出命令"O

UT"は、このうち破線で囲まれた部分の取り出しを要求するものである。

【0093】また、本実施例では、反復複写命令"RP"と挿入置換命令"IN"を付与した構造化文書を雛型文書とする。本実施例において使用する第1の雛型文書のHTMLファイルを以下に示しておく。

【0094】

【数10】
```
<HTML>
<HEAD>
<TITLE> Template1 </TITLE>
</HEAD>
<BODY BGCOLOR="#404040">
<H1>進捗サマリ</H1>
<TABLE BORDER="1">
<TR><TH>チーム</TH><TH>概況</TH></TR><!--#RP グループ-->
<TR>
<TH>
<!--#IN グループ-->
ここにグループ名が置換挿入される
<!--#/IN-->
</TH>
<TD>
<!--#IN サマリ-->
ここに各グループの概況が置換挿入される
<!--#/IN-->
</TD>
</TR>
<!--#/RP-->
</TABLE>
</BODY>
</HTML>
```

【0095】第1の雛型文書を解析木の形式で表現すると図5のようになる。

【0096】また、本実施例において使用する第2の雛型文書のHTMLファイルを以下に示しておく。

【0097】

【数11】
```
<HTML>
<HEAD>
<TITLE> Template2</TITLE>
</HEAD>
<BODY>
<H1>進捗サマリ</H1>
<TABLE BORDER="1">
<TR>
<TH>チーム</TH>
<!--#RP グループ-->
<TH>
<!--#IN グループ-->
ここにグループ名が置換挿入される
<!--#/IN-->
</TH>
<!--#/RP-->
</TR>
<TR>
<TH>概況</TH>
<!--#RP サマリ-->
<TD ALIGN="LEFT">
<!--#IN サマリ-->
ここに各グループの概況が置換挿入される
<!--#/IN-->
</TD>
<!--#/RP-->
</TR>
</TABLE>
</BODY>
</HTML>
```

【0098】第2の雛型文書を解析木の形式で表現すると図6のようになる。

【0099】雛型文書において、LABELは、抽出命令と、反復複写命令または挿入置換命令の対応を指定するのに用いる。

【0100】特定のLABEL引数が指定された抽出命令の数によって、同じラベルを有する反復複写命令の反復回数を定める。[数10]で示した第1の雛型文書の場合には、原料文書におけるLABEL引数「グループ」のみ反復複写される。これに対し、[数11]で示した第2の雛型文書の場合には、原料文書におけるLABEL引数「グループ」及び「サマリー」の双方が反復複写される。

【0101】また、特定のLABELが指定された抽出命令によって抽出された文書部品は、同じLABEL引数を有する挿入置換命令によって挿入または置換挿入される。[数10]及び[数11]で示した各雛型文書の場合には、原料文書におけるLABEL引数「グループ」及び「サマリー」の双方に関して挿入又は置換挿入される。

【0102】また、すべてのLABEL文字列が同一であるときは、LABEL文字列を省略したのと同等の効果をもたらす。

【0103】本実施例において、各命令は、HTMLファイルの注釈(コメント文)として構成される。したがって、HTMLブラウザを始めとするHTMLを取り扱う標準的なアプリケーション上では、これら各命令は無視されるので、文書の内容を表示したり処理するアプリケーションに対して実質的に影響を及ぼさない。各命令は、後続の命令分離手段によって各HTMLで記述された文書から分離され、取り出される。

【0104】命令付与手段は、本実施例では、HTML

ファイルを編集可能なエディタとして構成される。各命令は、単にHTMLファイル中に挿入された注釈タグなので、通常のテキスト・エディタあるいはHTMLエディタ上の編集作業を介して容易に付与することができる。例えば、ユーザは、テキスト・エディタを用いて、HTMLファイル中で取り出したい文書部品の前後それぞれに、抽出命令開始に相当する注釈タグ<!--#OUT LABEL-->、及び、抽出命令終了に相当する注釈タグ<!--#/OUT-->をマニュアルで挿入することによって、命令を付与することができる。以下、<!--#OUT LABEL-->を抽出命令開始タグと呼び、<!--#/OUT-->を抽出命令終了タグと呼ぶことにする。これら挿入される命令自体は、手続きの体裁をなしていないという点を充分理解されたい。なお、ここで用いられるテキスト・エディタは、一般的なエディタでよく、本明細書中ではこれ以上説明しない。

【0105】また、別の実現方法としては、計算機システム上で文書の構造を自動的に解析して、特定のマークアップで挟まれた文書部品の前後に、同様な各命令を挿入することも可能である。すなわち、特開平6-52161号公報（前述）において開示された方式を用いて構造を指定した検索を行ない、所望の文書部品の前後に所望の各命令の開始タグと終了タグを文書に流し込み合成を行なうことによっても、文書に対して命令を付与することができる。また、通常のテキスト処理によっても可能である。以下にperl言語のスクリプトで記述した例を示す（perlは米NetLabs社のLarry Wall氏が開発したインタプリタ言語であり、主としてUNIX系のOS上で使用される）。

【0106】

【数12】

```
$htmlfile = "sample.html";
open(HTML, "$htmlfile") || die "Can't open $htmlfile.";

while(<HTML>) {
   s/<TABLE[^>]*>/<!--#OUT LABEL-->$&/gi;
   s/<¥/TABLE[^>]*>/$&<!--#¥/OUT-->/gi;
   print;
}
```

【0107】上記の例では、正規表現によるパターン・マッチを用いて文書部品の検索を行なう。前記スクリプトは、HTMLファイルを入力して、<TABLE>タグを検索してその直前に<!--#OUT LABEL-->を挿入するとともに、</TABLE>タグを検索してその直後に<!--#/OUT-->を挿入して、結果を出力する。

【0108】次いで、入力手段と出力手段について説明する。

【0109】入力手段は、ファイル名を指定して、ファイル・サーバからHTMLファイルを入力する。ファイル・サーバは例えばWWW（World Wide Web）サーバである。WWW空間上では、URL（Uniform Resource Locator）をHTMLファイルの名前としてアクセスすることができる。

【0110】入力手段は、ネットワークに接続されたWWWサーバにネットワーク・コネクションを構成しており、HTTP（HyperText Transfer Protocol）として知られるプロトコルでHTMLファイルを転送する。すなわち入力手段は、HTMLファイル名を含むHTTPリクエストをWWWサーバに送信するとともに、WWWサーバからはHTMLファイルを含むHTTPレスポンスを受信する。より具体的には、所定のポート番号で待ち受けるWWWサーバに対して、TCP/IP（Transmission Control Protocol/Internet Protocol）コネクションを形成し、HTTPのリクエストを入力する。HTTPリクエストには、URLとして与えられたファイル識別子が含まれている。これに対し、WWWサーバは、URLに基づいて、HTMLファイルをローカルなファイル・システムから読み出すか、所定のプログラムを起動してHTMLファイルを合成するか、又は、他のWWWサーバからHTMLファイルを入力するかして、HTTPリクエストで指定されたURLに対するHTMLファイルを用意し、これをHTTPレスポンスに含めて入力手段に出力する。

【0111】このような入力手段の実装は、例えば米サンマイクロシステムズ社のJDK（Java Development Kit）に含まれる、java.net.URL、及び、Java.net.URLConnectionクラスの実装を用いればよい。

【0112】出力手段は、構造化文書をHTML形式に変換したものを、HTTPレスポンスの本体すなわちbody部として、ネットワークを経由して出力する。すなわち、本実施例の構造化文書処理システムは、クライ

アントからネットワークを経由して適当なリクエストを受信し、ファイル・サーバから原料文書と雛型文書を入力して処理を行ない、処理結果をレスポンスとしてクライアントに出力する構造化文書処理サーバとして実現される。特に、本実施例に係る構造化文書処理サーバは、ＨＴＴＰリクエストを入力して所定の構造化文書の処理を行ない、ＨＴＴＰレスポンスを出力するＨＴＴＰサーバとして実現する。

【０１１３】出力手段に送られる構造化文書は、該構造化文書の構文解析木とする。構文解析木については後述する。構文解析木をＨＴＭＬに変換するには、該構文解析木を深さ優先順に走査して、開始タグ、テキストコンテント、及び、終了タグを出力する。走査において訪れたノードがテキスト・コンテントを保持するノードである場合には、テキスト文字列を出力する。そうでなければ、開始タグ（例えばＴＡＢＬＥタグなら”＜ＴＡＢＬＥ＞”）を出力し、次に子ノードを順に走査し、各子ノードについて再帰的に開始タグと終了タグとテキスト・コンテントをＨＴＭＬとして出力する。子ノードの出力の後に、該ノードを上向きに訪れるときに終了タグ（例えばＴＢＡＬＥタグなら”＜／ＴＡＢＬＥ＞”）を出力する。ＨＴＭＬをＨＴＴＰレスポンスに含めて出力することは、通常のＨＴＰＰサーバの場合と同様なので、本明細書中ではこれ以上説明しない。また、ＨＴＴＰリクエストを入力する部分については、本発明の要旨とは直接関係しないので、説明を省略する。このようなＨＴＴＰサーバの実現例としては、Ａｐａｃｈｅ　ＷＷＷサーバがよく知られている。

【０１１４】次いで、解析手段と分離手段について説明する。

【０１１５】解析手段は、テキスト形式で記述されたＨＴＭＬファイルを入力し、ＨＴＭＬの文法（構文）に従って解析し、構文解析木を構成するＨＴＭＬパーザである。本発明を実現する上で、従来技術と同様のＨＴＭＬパーザを利用することができるので、本明細書中では詳しく説明しない。（ＨＴＭＬパーザの実現例については、Ｃｏｒｐｏｒａｔｉｏｎ　ｆｏｒ　Ｎａｔｉｏｎａｌ　ＲｅｓｅａｒｃｈＩｎｉｔｉａｔｉｖｅｓ，「Ｐｙｔｈｏｎ Ｌｉｂｒａｒｙ Ｒｅｆｅｒｅｎｃｅ Ｒｅａｌｅａｓｅ　１．５．２」１２．２節 ｈｔｍｌｌｉｂ － Ａ ｐａｒｓｅｒ ｆｏｒ ＨＴＭＬ ｄｏｃｕｍｅｎｔｓなどに記載されている。また、構文解析の方式については、Ａ．Ｖ．エイホ他、コンパイラ１サイエンス社４章に詳述されている）。

【０１１６】本実施例で用いるＨＴＭＬパーザは、テキスト形式のＨＴＭＬファイルを入力すると、木構造のデータを出力する。木構造の各ノードはメモリに一時格納される。各ノードは、ＨＴＭＬタグ名または’ＴＥＸＴ’をノード名として属性に持つ。この場合、各ノードがＨＴＭＬタグのマークアップ又はテキスト・コンテンツに対応する。また、ＨＴＭＬタグの属性（Ａｔｔｒｉｂｕｔｅ）を属性に持つ。さらに子ノードへのポインタの配列を持つ。この配列は、第１の要素、第２の要素…の順に、第２子ノード、第２子ノード…へのポインタが格納される。解析木中の任意のノードを指定することで、該ノード以下のサブツリーが文書部品として定まる。（木構造の実現については、例えば「アルゴリズムとデータ構造」岩波書店、ｐ．４８～ｐ．５４に詳解されている。）

【０１１７】第１の原料文書、第２の原料文書、第１の雛型文書、第２の雛型文書の構文解析木（以下、「文書解析木」と呼ぶ）は、それぞれ図３、図４、図５、図６に示してある通りである。

【０１１８】次いで、解析手段としてのＨＴＭＬパーザは、構文解析木を行きがけ順（ｐｒｅｆｉｘ　ｏｒｄｅｒ）に巡回し、ノード名が注釈タグであって、注釈タグの中に記述されている注釈文字列が、次の規則のいずれかに該当するか否かを、文字列のパターン・マッチによって調べる。但し、「→」記号の左辺がパターンであり、右辺が各命令のいずれかである。また、＜空白＞は１以上の空白記号を意味し、＜ＬＡＢＥＬ文字列＞は、空白を含まない任意の文字列を意味するものとする（木構造を行きがけ順に巡回する方法については、例えば「アルゴリズムとデータ構造」　岩波書店、ｐ．５２～ｐ．５３に木構造の前順走査として記載されている）。

【０１１９】

【数１３】

”＃ＯＵＴ＜空白＞＜ＬＡＢＥＬ文字列＞”　→　抽出命令開始タグ
”＃／ＯＵＴ＜空白＞＜ＬＡＢＥＬ文字列＞”　→　抽出命令終了タグ
”＃ＲＰ＜空白＞＜ＬＡＢＥＬ文字列＞”　→　反復複写命令開始タグ
”＃／ＲＰ＜空白＞＜ＬＡＢＥＬ文字列＞”　→　反復複写命令終了タグ
”＃ＩＮ＜空白＞＜ＬＡＢＥＬ文字列＞”　→　挿入置換命令開始タグ
”＃／ＩＮ＜空白＞＜ＬＡＢＥＬ文字列＞”　→　挿入置換命令終了タグ

【０１２０】このようなパターン・マッチは、正規表現による文字列マッチを用いて実現可能である。Ｐｅｒｌ言語で利用可能なパターン・マッチによって各命令の開始タグと終了タグを抽出するプログラムの例を、以下に示しておく。

【０１２１】

【数14】
```
$htmlfile = "Template2.html";
open(HTML, "$htmlfile") ||
die "Can't open $htmlfile.";
while(<HTML>){
@_=split('>', $_);
foreach $_ (@_){
$value='';
$value="抽出命令開始タグ／". $1 if /<!--#OUT_([^¥s]+)¥s*--/;
$value="反復複写命令開始タグ／". $1 if /<!--#RP_([^¥s]+)¥s*--/;
$value="挿入置換命令開始タグ／". $1 if /<!--#IN_([^¥s]+)¥s*--/;
$value="抽出命令終了タグ／". $1 if /<!--#¥／OUT_([^¥s]+)¥s*--／;
$value="反復複写命令終了タグ／". $1 if /<!--#¥／RP_([^¥s]+)¥s*--／;
$value="挿入置換命令終了タグ／". $1 if /<!--#¥／IN_([^¥s]+)¥s*--／;
print "$value¥n" if $value ne '';
}
}
```

【0122】文書解析木中のアドレスは、文書部品のノードの位置を解析木のルート・ノードからのパスを示したベクトルによって表される。ベクトルの各要素はパス上に存在するノードに対応する。深さが3の位置にある文書部品のアドレスは、3次元のベクトルで表現され、ベクトルの各要素は、それぞれのノードの親に対して（長子を0と数えて、）何番目の子であるかを表す数とする。例えば、[0 12]というアドレス表現は、ルートの第2子の第3子に位置するノード以下のサブツリーで表される文書部品のアドレスである。所望のノードのアドレスが指定されれば、次のステップではルート・ノードから該所望のノードを対象ノードとする。

【0123】

【数15】Step1：対象ノードをルート・ノードに設定し、「深さ」を1に設定する。Step2～Step4をベクトルの次元数－1だけ繰り返す

Step2： アドレス・ベクトルの各要素のうち「深さ」番目の番号（iとする）を取り出す。（ただし先頭を0番目とする）

Step：3対象ノードが保持する子ノード配列から「i」番目を取り出し、新たに対象ノードとする（ただし配列の先頭を0番目とする）。

Step4：「深さ」に1を加える。

【0124】このような解析木のアドレス表現について、図5を参照しながら説明する。同図においては、反復複写命令[0 1 1 1]が付与された文書部品は、[01 1 2]のノード以下のサブツリーであり、第1の挿入置換命令[0 11 2 0 0]が付与された文書部品は、[0 1 1 2 0 1]のノード以下のサブツリーであり、第2の挿入置換命令[0 1 1 2 1 0]が付与された文書部品は、[0 1 1 2 1 1]のノード以下のサブツリーである。

【0125】文書部品IDは、文書部品を識別するためのもので、このIDを指定して文書部品を取り出すことができる。本実施例では、文書解析木IDと文書部品のアドレスの組で文書部品IDを表すものとする。文書解析木のIDは、例えば、構成した解析木の順に割り振った番号としてもよいし、文書解析木を格納したメモリ上のアドレスとして、例えば文書解析木のルート・ノードへのポインタでもよい。

【0126】次いで、命令分離手段について説明する。本実施例の命令分離手段は、文書解析木を行きがけ順に巡回し、各命令に対するオペコードと、ラベル文字列と、各命令が付与された文書部品のIDを前記オペコードに対するオペランドとする3項目からなる命令列を出力する。オペコードは、抽出命令はOUT、反復複写命令はRP、挿入置換命令はINとする。文書解析木のIDは、第1の原料文書を1、第2の原料文書を2、第1の雛型文書を3、第2の雛型文書を4とする。それぞれの文書から分離した命令列を以下に示す。

【0127】

【数16】

第1の原料文書から分離した命令列

オペコード	ラベル	文書部品ID
OUT	グループ	<0, [0 1 0 1]>
OUT	サマリ	<0, [0 1 4]>

【0128】

【数17】

第2の原料文書から分離した命令列

オペコード	ラベル	文書部品ID
OUT	グループ	<1, [0 1 0 2]>

	ＯＵＴ	サマリ

<1, [0 1 4]>

【０１２９】

【数１８】

第１の雑型文書から分離した命令列

オペコード	ラベル
ＲＰ	グループ
ＩＮ	グループ
ＩＮ	サマリ

文書部品ＩＤ

<2, [0 1 1 2]>
<2, [0 1 1 2 0 1]>
<2, [0 1 1 2 1 1]>

【０１３０】

【数１９】

第２の雑型文書から分離した命令列

オペコード	ラベル
ＲＰ	グループ
ＩＮ	グループ
ＲＰ	サマリ
ＩＮ	サマリ

文書部品ＩＤ

<3, [0 1 1 0 1]>, <3, [0 1 1 0 2 1]>
<3, [0 1 1 1 1]>, <3, [0 1 1 1 2 1]>

【０１３１】次いで、文書処理記述合成手段について説明する。文書処理記述合成手段は、複数の原料文書または雑型文書から分離した命令列を入力し、命令列を併合し、順序の変更をし、変換と合成を行なって文書処理記述を出力する。

【０１３２】ここで、原料文書１と原料文書２と雑型文書１のそれぞれから分離した命令列から文書処理記述を構成する処理を例にとって、文書処理記述合成手段の動作について説明する。文書処理記述合成手段は、以下に示す処理Ａ～処理Ｃを順に行なう。

【０１３３】■処理Ａ：　ＯＵＴの計数

原料文書１と原料文書２のそれぞれから分離した命令列に含まれるＯＵＴの個数をラベル文字列毎に数える。これは、以下に示すように、命令列を走査して、オペコードがＯＵＴである命令について、ラベル文字列をキーとして、整数値を値とする連想配列をインクリメントすることで実現できる。走査が終了した後で、ラベル文字列をキーとしてＯＵＴ命令の個数を各ラベル文字列ごとに取り出すことができる。

【０１３４】

【数２０】すべて原料文書のすべての命令列について

Ｓｔｅｐ１１：オペコードがＯＵＴならＳｔｅｐ１２を実行。そうでなければ次の命令を調べる

Ｓｔｅｐ１２：オペランドのラベル文字列をキーとして、連想配列を調べる。連想配列が未登録であれば、値として１を設定する。連想配列が登録されていれば、値に１を加える。

Ｓｔｅｐ１３：次の命令についてＳｔｅｐ１１を実施。

【０１３５】■処理Ｂ：　ＲＰによる各命令の複写

雑型文書１の命令列において、命令列中にＲＰが含まれていた場合、すべてのＲＰのオペランドの文書部品中に含まれるＩＮとＯＵＴとＲＰを、原料文書のＯＵＴ命令の数に従って複写する。この複写は１つの雑型文書について、以下に示す手順Ｓｔｅｐ２１～Ｓｔｅｐ２６を実行することで行われる。

【０１３６】

【数２１】Ｓｔｅｐ２１：ＲＰリストを作成する。

Ｓｔｅｐ２２：ＲＰリストから、対象とするＲＰ（反復複写命令）を順に変更しつつ各ＲＰについてＳｔｅｐ２３以降の処理を実行する。

Ｓｔｅｐ２３：対象ＲＰのオペランドのラベル文字列を取り出し、上記の処理Ａで作成した連想配列から値を取り出し、反復複写回数ｔとする。ｔが２以上ならＳｔｅｐ２４以降の手順を実行する。

Ｓｔｅｐ２４：反復複写回数ｔをＲＰ命令のオペランドのラベル文字列に代えて、文書処理記述に出力する。

Ｓｔｅｐ２５：オペランドから、文書部品のＩＤを取り出して、第１のアドレスを取り出す。第１のアドレスの次元数（要素の数）をｋとする。

Ｓｔｅｐ２６：　命令列の各命令のうち、Ｓｔｅｐ２２で取り出した第１のアドレスより次元数が大きく、かつ、先頭ｋ個が一致するアドレスをオペランドに持つ命令を取り出す。例えば、[０ １ ２ ３ ４]と[０ １ ２]は、先頭３個が一致するので、[０ １ ２ ３ ４]は[０ １ ２]のノードの子孫である。また、[０ １ １ ０ ２]と[０ １] は先頭２個が一致する。[０ １ １ ０２]は[０ １]のノードの子孫である。このようなチェックは、２つの配列間で要素同士を比較することで容易に実現できる。

Ｓｔｅｐ２７：Ｓｔｅｐ２４で取り出した命令を複写し、複写された命令のオペランドのアドレス（第２のアドレス）を次のように書き換える。

第１のアドレス　＝　[Ａ１ Ａ２ Ａ３ … Ａｋ－１ Ａｋ]　（ＲＰ命令のオペランドの文書部品のアドレス）

第２のアドレス　＝　[Ａ１ Ａ２ Ａ３ … Ａｋ－１ Ａｋ Ｎ１ Ｎ２ Ｎ３…]

書き換え後のアドレス　＝[Ａ１ Ａ２ Ａ３ …

Ak－1　Ak＋iN1　N2　N3…]
但し、iは、複写の度に増加する整数、すなわち、1，2，…，t－1とする。また、RP命令のオペランドに複数の連続した文書部品が指定されている場合は、指定されている文書部品の数をn個とすると、iは、複写の度にnずつ増加する。例えばn－2の場合、iは、2，4，…，(t－1)×nとする。
【0137】複写された命令以外のすべての命令について、オペランドの文書部品のアドレス（第3のアドレス）の先頭k－1個が一致し、かつ、k番目がAkより大きいものについて、アドレスの付け替えを行なう。
【0138】
【数22】第1のアドレス ＝ [A1　A2　A3 … Ak－1　Ak] （RP命令のオペランドの文書部品のアドレス）
第3のアドレス ＝ [A1　A2　A3 … Ak－1　B　N1N2　N3…]
書き換え後のアドレス ＝[A1　A2　A3 … Ak－1　B+t－1　N1　N2　N3…]
但し、B＞ Akとし複写する度に増加する整数、すなわち、1、2，…，t－1とする。
【0139】この付け替えは、反復複写によるアドレスのずれを補正する処理となる。複写した命令がRPの場合は、Step21で作成したRPリストに追加する。
【0140】
【数23】第1の雛型文書から分離した命令列に処理Bを施した結果えられる命令列

オペコード	ラベル	文書部品ID
RP	2	<2，[0　1　1　2]>
IN	グループ	<2，[0　1　1　2　0　1]>
IN	サマリ	<2，[0　1　1　2　1　1]>
IN	グループ	<2，[0　1　1　3　0　1]>
IN	サマリ	<2，[0　1　1　3　1　1]>

【0141】
【数24】第2の雛型文書から分離した命令列に処理Bを施した結果えられる命令列

オペコード	ラベル	文書部品ID
RP	2	<3，[0　1　1　0　1]>，<3，[0　1　1　0　2]>
IN	グループ	<3，[0　1　1　0　2　1]>
IN	グループ	<3，[0　1　1　0　4　1]>
RP	サマリ	<3，[0　1　1　1　1]>，<3，[0　1　1　1　2]>
IN	サマリ	<3，[0　1　1　1　2　1]>
IN	サマリ	<3，[0　1　1　1　4　1]>

【0142】■処理C：　命令の合成
すべての原料文書のOUTと、雛型文書のINとを、ラベル文字列が一致するものどうしでペアを構成して各ペアから新しい命令を合成する。この動作手順について以下に説明する。
【0143】本実施例では、例えば”OUT　LABEL　文書部品1”と”INLABEL　文書部品2”というペアから”FILL　文書部品1　文書部品2”を合成する。FILLは、文書部品1を文書部品2と置換する命令を表す。この動作によって原料文書に出現する文書部品の出現順にかかわらず、雛型文書の中のユーザが所望する位置に、前記原料文書の文書部品を置換または挿入することが可能になる。
【0144】
【数25】Step31：　原料文書のすべての命令列を走査し、ラベル文字列ごとに命令の配列を作成する。この配列はOUT配列と呼ぶ。OUT配列は、ラベル文字列をキーとする連想配列に格納しておく。
Step32：　雛型文書のすべての命令列を走査し、ラベル文字列ごとに命令の配列を作成する。この配列はIN配列と呼ぶ。IN配列は、ラベル文字列を属性として保持する。
Step33：すべてのIN配列について、Step34以降を実行する。
Step34：IN配列の属性であるラベル文字列から、一致する文字列をオペランドに持つOUT配列を取り出す。
Step35：IN配列のすべての要素に対して、0番目から順にm番目を対象要素としてStep36以降を実行する。
Step36：OUT配列のm番目の要素として格納されたOUT命令のオペランドの文書部品のIDである文書部品1を取り出す。IN配列のm番目の要素として格納されたIN命令のオペランドの文書部品のIDである文書部品2を取り出す。
Step37：文書処理記述に次の命令を追加する。この例では、オペコードはFILLで、オペランドは文書部品1と文書部品2である

FILL　文書部品1　文書部品2

【0145】また、文書部品に含まれるコンテンツ、例えば「社員番号」を取り出して、社員番号順にOUT命令を整列してOUT配列を構成し、雛型文書中のIN命令の出現順に構成したIN配列に対応付けて、FILLを構成することも可能である。このように、原料文書のOUT命令で抽出される文書部品と、IN命令で指定される挿入置換の場所との対応付けは、ラベル文字列の対応付けや、コンテンツの内容による整列だけにとどまらず、さまざまな方法が実施可能である。例えば、テーブルを検索して、順序を決めることもできるし、どのようなサーバから得られた原料文書か（例えばホスト名）によって、挿入置換場所を変更することも可能である。この場合は、例えばIN命令のラベル文字列をホスト名にしておき、OUT命令のオペランドの文書IDと、ホスト名の対応表を基に、同様の処理をすることで実現できる。

【0146】これらの処理によって文書処理記述が合成される。文書処理記述の例を以下に示しておく。

【0147】

【数26】

※第1の原料文書と第2の原料文書と第1の雛型文書から合成された文書処理記述例1

```
RP      2          2,[0 1 1 2]
FILL    0,[0 1 0 1]   2,[0 1 1 2 0 1]
FILL    1,[0 1 0 2]   2,[0 1 1 3 0 1]
FILL    0,[0 1 4]     2,[0 1 1 2 1 1]
FILL    1,[0 1 4]     2,[0 1 1 3 1 1]
```

【0148】

【数27】

※第1の原料文書と第2の原料文書と第2の雛型文書から合成された文書処理記述の例2

```
RP      2          3,[0 1 1 0 1]    3,[0 1
  1 0 2]
RP      2          3,[0 1 1 1 1]    3,[0 1
  1 1 2]
FILL    0,[0 1 0 1]   3,[0 1 1 0 2 1]
FILL    1,[0 1 0 2]   3,[0 1 1 0 3 1]
FILL    0,[0 1 3]     3,[0 1 1 1 2 1]
FILL    1,[0 1 3]     3,[0 1 1 1 3 1]
```

【0149】上記の例では、1つのRP命令は、2つの連続する文書部品に付与されているので、RP命令のオペランドは2個指定されていることに注意する。また、FILL命令を合成するのではなく、文書部品を格納する一時領域を指定するラベルを用意することで、次のような文書合成記述を出力してもよい。

【0150】

【数28】

※原料文書1と原料文書2と雛型文書2から合成された文書処理記述の例3

```
RP      2          2,[0 1 1 2]
OUT     0,[0 1 0 1]   tmp
IN      tmp        2,[0 1 1 2 0 1]
OUT     0,[0 1 4]     tmp
IN      tmp        2,[0 1 1 2 1 1]
OUT     1,[0 1 0 2]   tmp
IN      tmp        2,[0 1 1 3 0 1]
OUT     1,[0 1 4]     tmp
IN      tmp        2,[0 1 1 3 1 1]
```

【0151】次に、インタプリタと、抽出手段と、反復複写手段と、挿入置換手段の動作について説明する。

【0152】上述した文書処理記述がインタプリタに入力されると、インタプリタは、文書処理記述の先頭から順に走査し、オペコードを判定して、抽出手段、反復複写手段、又は挿入置換手段のいずれかに、オペランドで指定された文書部品IDを入力する。OUT命令には抽出手段が、RP命令には反復複写手段が、IN命令には挿入置換手段が、それぞれ対応するものとする。挿入置換手段には、文書部品のIDに加えて、挿入または置換する文書部品を入力する。FILLの場合には、第1のオペランドで指定された文書部品IDが抽出手段に入力され、その結果として得られた文書部品と、第2のオペランドで指定された文書部品IDを挿入置換手段に入力

する。

【０１５３】本実施例のインタプリタは、命令記述を読み込んで、各命令を１ステップ毎に順次解釈して所定の処理を実行する処理を行なうような通常のインタプリタでよい。かかるタイプのインタプリタの実現は当業界において既に周知なので、本明細書では説明を省略する（例えば、インタプリタの実現方法は、滝口政光、「作りながら学ぶコンパイラ／インタプリタ」（ＣＱ出版株式会社）に詳しく説明されている）。本実施例では、文書処理記述内の命令ステップの順に、各オペコードに対して所定の処理手段が選択され、各処理手段にオペランドとして文書部品のＩＤが入力される。

【０１５４】抽出手段は、入力された文書部品のＩＤから、指定された文書解析木のノードを見つけて、コピーした文書部品を返す。コピーは該ノード以下のすべてのノードを行きがけ順に巡回し、各ノードごとにコピーするものとする。コピーした文書部品は、メモリ上に格納されてポインタが返されるように実装してもよい．先の文書処理記述例３の一時格納領域として指定したｔｍｐは、このようなコピーした文書部品を格納できるように実装してもよいし、前記ポインタであってもよい。

【０１５５】反復複写手段は、入力された文書部品のＩＤから、指定された文書解析木のノードを見つけて、該文書解析木上で反復複写する。（反復複写した結果、アドレスがずれるノードがあるため、先の処理ＢのＳｔｅｐ２４で、オペランドに指定された文書部品のアドレスの付け替えを行なっていることに注意する。）

【０１５６】図７には、雛型文書１に対して、文書処理記述例１のＲＰ命令で反復複写した結果の文書解析木の例を示している。また、図８には、雛型文書２に対して、文書処理記述例２のＲＰ命令で反復複写した結果の文書解析木の例を示している。太枠で囲まれた部分が反復複写された部分を表している。

【０１５７】挿入置換手段には、挿入又は置換する第１の文書部品（又はそのＩＤ）に加えて、挿入置換をする個所を示す第２の文書部品のＩＤを入力する。入力された第２の文書部品のＩＤから、指定された文書解析木のノードを見つけて、前（兄ノード）又は後ろ（弟ノード）に、指定した第１の文書部品のノードを挿入する。置換動作の場合は、第２の文書部品として指定されたノードを削除して、その代りに指定した別の文書部品のノードを挿入する。第１又は第２の文書部品の代りに複数の文書部品を指定して、挿入又は置換するように実現してもよい。抽出手段、反復複写手段、挿入置換手段の各々は、上記のように単に文書解析木のツリー構造の変形や変換を行なうものであり、その実現は当業界において既に周知なので、本発明ではこれ以上説明しない（木構造に対する要素の挿入や削除については、例えば「アルゴリズムとデータ構造」（岩波書店）、２．４節に説明されている）。

【０１５８】インタプリタによって、文書処理記述が解釈され雛型文書から構成された解析木が処理され、文書が合成される。図９には、第１の原料文書と第２の原料文書と第１の雛型文書から合成した文書の解析木を示している。同図において、網掛けを付されたノードが、置換された文書部品に相当する。合成結果のＨＴＭＬファイルを以下に示しておく。

【０１５９】

【数２９】

```
<HTML>
<HEAD>
<TITLE> Template1 </TITLE>
</HEAD>
<BODY BGCOLOR="#404040">
   <H1>進捗サマリ</H1>
   <TABLE BORDER="1">
          <TR><TH>チーム</TH><TH>概況</TH><
/TR>
<!--#RP グループ-->
      <TR>
          <TH>
<!--#IN グループ-->Aチーム<!--#/IN-->
          </TH>
          <TD>
<!--#IN サマリー--><UL>
                <LI> 項目１の進捗６０％ 計画どおり </L
I>
                <LI> 項目２の進捗３０％ 遅れ： ３日 </
LI>
```

```
                    <LI> 項目3の進捗70% 予定外の問題発生、
計画見直しが必要 </LI>
                </UL><!--#/IN-->
                </TD>
            </TR>
            <TR>
                <TH>
<!--#IN グループ-->Bチーム<!--#/IN-->
                </TH>
                <TD>
<!--#IN サマリー--><TABLE BORDER="0" CELL
SPACING="0" CELLPADDING="1">
                    <TR><TH>項目</TH><TH>進捗</T
H><TH ALIGN="LEFT"">評価</TH></TR>
                    <TR><TH>1</TH><TD>20%</T
D><TD>やや計画おくれ(1日)</TD></TR>
                    <TR><TH>2</TH><TD>40%</T
D><TD>遅れなし</TD></TR>
                    <TR><TH>3</TH><TD>60%</T
D><TD>進み: 3日</TD></TR>
                </TABLE><!--#/IN-->
                </TD>
            </TR>
<!--#/RP-->
</TABLE>
</BODY>
</HTML>
```

【0160】上記のHTMLファイルを標準的なHTMLブラウザで表示した結果を図20に示しておく。

【0161】同様に、第1の原料文書と第2の原料文書と第2の雛型文書から合成した文書のHTMLファイルを以下に示す。

【0162】

【数30】

```
<HTML>
<HEAD>
<TITLE> Template2 </TITLE>
</HEAD>
  <BODY>
  <H1>進捗サマリ</H1>
  <TABLE BORDER="1">
            <TR>
            <TH>チーム</TH>
<!--#RP グループ-->
            <TH>
<!--#IN グループ-->Aチーム<!--#/IN-->
            </TH>
            <TH>
<!--#IN グループ-->Bチーム<!--#/IN-->
            </TH>
<!--#/RP-->
        </TR>
        <TR>
```

```
            <TH>概況</TH>
<!−−#RP サマリー−>
            <TD ALIGN=" LEFT" >
<!−−#IN サマリー−>
            <UL>
                  <LI> 項目1の進捗60% 計画どおり </L
I>
                  <LI> 項目2の進捗30% 遅れ: 3日 </
LI>
                  <LI> 項目3の進捗70% 予定外の問題発生、
計画見直しが必要 </LI>
            </UL><!−−#/IN−−>
            </TD>
            <TD ALIGN=" LEFT" >
<!−−#IN サマリー−>
                  <TABLE BORDER=" 0" CELLSP
ACING=" 0" CELLPADDING=" 1" >
                  <TR><TH>項目</TH><TH>進捗</T
H><TH ALIGN=" LEFT" " >評価</TH></TR>
                  <TR><TH>1</TH><TD>20%</T
D><TD>やや計画おくれ(1日)</TD></TR>
                  <TR><TH>2</TH><TD>40%</T
D><TD>遅れなし</TD></TR>
                  <TR><TH>3</TH><TD>60%</T
D><TD>進み: 3日</TD></TR>
                  </TABLE><!−−#/IN−−>
            </TD>
<!−−#/RP−−>
      </TR>
</TABLE>
</BODY>
</HTML>
```

【0163】上記のHTMLファイルを標準的なHTMLブラウザで表示した結果を図21に示しておくが、これは図20に示した文書の合成結果とは明らかに相違する。すなわち、同じ原料文書1及び原料文書2に対して本実施例に係る構造化文書処理を実行した場合であっても、適用した雛型文書が相違すれば合成文書も異なるという点を理解されたい。

【0164】以上説明したように、本実施例に係る構造化文書処理システムは、命令付与手段によって抽出命令が付与された第1の構造化文書と、同じく命令付与手段によって反復複写命令と挿入置換命令が付与された第2の構造化文書とを入力とし、解析手段によって各文書毎に文書解析木を構成して、命令分離手段によって各文書から命令列を取り出し、取り出した複数の該命令列を文書処理記述合成手段によって併合、整列、及び変換を行なって文書処理記述を合成し、インタプリタによって文書処理記述を順に走査し、文書処理記述中に含まれる各命令に従って抽出手段によって第1の文書部品を第1の文書から抽出し、反復複写手段によって第2の構造化文書の文書部品を第1の文書部品の数に依存して定まる回数だけ反復複写し、挿入置換手段によって第2の構造化文書に第1の文書部品を挿入又は置換することで、構造化文書の合成を行なうことができる。

【0165】本実施例に係る構造化文書処理システムによれば、

1. 原料文書の構造を変更した際に、これを処理するための別に管理されるスクリプトを用意しなくてよい。
2. 構造が異なる複数の原料文書を混在させて文書部品を取り出すには、各文書毎に文書部品に抽出命令を付与することで行なうので、文書毎の抽出の処理を個別に指定しなくてよい。同様に、文書部品を挿入又は置換する処理の指定も不要である。また、ユーザは、文書の構造が変わる度に適切なスクリプトを指定する必要がない。
3. 文書部品は、所望の命令を元の文書中に直接的に付与することと、取り出した文書部品と挿入置換される文

書部品の条件を指定すること（例えばラベル文字列が一致するなど）によって作成することができる。また、各ユーザがそれぞれの目的に応じて雛型文書をデザイン（作成）したり、所望の処理を行なう命令が組み込まれた雛型文書を選択して、ユーザ自らが原料文書と組み合わせることで文書処理アプリケーションを構成することができる。このように手続き的でない、言い換えれば宣言的な方法で文書処理を構成することができるので、プログラミングの知識を充分に持たないユーザが広く文書処理を行なうことができる。すなわち、各ユーザがそれぞれの目的に合わせた構造化文書処理のアプリケーションを容易に作成できる文書処理システムが実現できる。

【０１６６】次いで、本発明の第２の実施例について説明する。

【０１６７】図１０には、本発明の第２の実施例に係る構造化文書処理システムの構成を模式的に示している。本実施例では、抽出命令に代えて属性抽出命令が、挿入置換命令に代えて属性置換命令が、それぞれ付与された構造化文書に対して文書処理を実行するものである。以下、第１の実施例との相違点を中心に本実施例に関して説明する。

【０１６８】命令付与手段は、第１の実施例と同様に、ＨＴＭＬエディタによって構成される。本実施例に係る命令付与手段は、入力される文書内に、抽出命令に代えて属性抽出命令を、挿入置換命令に代えて属性置換命令を、それぞれ埋め込む。各命令は、次のように記述される。すなわち、

【０１６９】

【数３１】・属性抽出命令

＜！−−＃ＧＥＴ　ＬＡＢＥＬ　タグ名−−＞

※ＨＴＭＬで記述された文書部品

＜！−−＃／ＧＥＴ−−＞

【０１７０】

【数３２】・属性置換命令

＜！−−＃ＰＵＴ　ＬＡＢＥＬ　タグ名　属性名−−＞

※ＨＴＭＬで記述された文書部品

＜！−−＃／ＰＵＴ−−＞

【０１７１】上記において、’ＧＥＴ’が属性抽出命令を、’ＰＵＴ’が属性置換命令を、それぞれ表わすものとする。ここで、＜！−−＃ＧＥＴ　ＬＡＢＥＬ　タグ名−−＞及び＜！−−＃／ＧＥＴ−−＞をそれぞれ属性抽出命令開始タグ及び属性抽出命令終了タグとし、同様に、＜！−−＃ＰＵＴ　ＬＡＢＥＬ　タグ名　属性名−−＞及び＜！−−＃／ＰＵＴ−−＞をそれぞれ属性置換命令開始タグ及び属性置換命令終了タグとする。

【０１７２】属性抽出命令を付与したＨＴＭＬファイルの例を以下に示す。本明細書中では、該ＨＴＭＬファイルを第３の原料文書とする。

【０１７３】

【数３３】

```
<HTML>
<HEAD>
<TITLE> Sample3 </TITLE>
</HEAD>
<BODY>
   <H2>リンク集</H2>
   <HR>
<!−−#GET リンク A−−>
      <A HREF="http://sample.com/" ta
rget="win0">リンク1</A>
<!−−#/GET−−>
<HR>
</BODY>
</HTML>
```

【０１７４】この第３の原料文書を標準的なＨＴＭＬブラウザで表示した様子を図２２に示しておく。

【０１７５】また、属性置換命令を付与したＨＴＭＬファイルの例を以下に示す。本明細書では、該ＨＴＭＬファイルを第３の雛型文書とする。

【０１７６】

【数３４】

```
<HTML>
<HEAD>
<TITLE> Template3 </TITLE>
</HEAD>
<BODY>
   <H1>資料集</H1>
   <HR>
```

```
   <IMG SRC="images/chart.gif" alt="チ
ャート" WIDTH="600" HEIGHT="200" BORDER
="0" USEMAP="#mapdata0">
   <MAP NAME="mapdata0">
<!--#PUT リンク AREA=HREF-->
   <AREA SHAPE="rect" ALT="リンク集" coor
ds=0,20,200,85 HREF="">
<!--#/PUT-->
</MAP>
<HR>
</BODY>
</HTML>
```

【0177】命令付与手段及び入力手段の構成及び動作特性は上記した第1の実施例と略同一なので、ここでは説明を省略する。

【0178】解析手段についても第1の実施例と同様で

"#GET<空白><LABEL文字列><空白><タグ名文字列>" → 属性抽出命令開始タグ

"#/GET" → 属性抽出命令終了タグ

"#PUT<空白><LABEL文字列><空白><タグ名文字列>=<属性名文字列>" → 属性置換命令開始タグ

"#/PUT" → 属性置換命令終了タグ

【0180】図11には第3の原料文書の解析木を、また、図12には第3の雛型文書の解析木をそれぞれ示している。

【0181】図11において、属性抽出命令[0 1 2]が付与された文書部品は[01 3]のノードである。同様に、図12において属性置換命令[0 1 3 0]が付与された文書部品は[0 1 3 1]のノードである。これら文書部品中において、属性抽出命令又は属性置換命令で指定されたタグ名のタグをそれぞれ属性抽出命令、属性置換命令の対象タグとする。本実施例では、属性抽出命令の対象タグは<A>タグであり、また、属性置換命令の対象タグは<AREA>タグである。文書部品中に複数の対象タグがあってもよい。対象タグの検索は、上記した第1の実施例の場合と同様に、ツリーを行きがけ順に巡回して行い、引数で指定された

第3の原料文書から分離した命令列

オペコード	ラベル	タグ名	文書部品ID
GET	リンク	A	<5, [0 1 3]>

【0185】

第3の雛型文書から分離した命令列

オペコード	ラベル	タグ名	属性名	文書部品ID
PUT	リンク	AREA	HREF	<6, [0 1 3 1]>

【0186】文書処理記述合成手段は、複数の原料文書または雛型文書から分離した命令列を入力し、命令列を併合し、順序の変更をし、変換と合成を行なって文書処理記述を出力する。以下で説明する点を除いては、文書処理記述合成手段は第1の実施例と同様である。

ある。但し、本実施例に係る解析手段は、注釈タグについては以下の規則に従うか否かを解析する。すなわち、

【0179】

【数35】

タグ名のタグを集めることができる。

【0182】命令分離手段もまた、第1の実施例のそれと略同一である。但し、第1の実施例とは相違して、オペコードは属性抽出命令であるGETと、属性挿入命令であるPUTとなる。また、属性抽出命令にはタグ名、属性挿入命令にはタグ名と属性名がそれぞれ該命令の引数として付与されているため、それらをオペランドとして出力する。

【0183】命令分離手段によって、第3の原料文書及び第3の雛型文書の各々から分離した命令列を以下に示す。但し、文書解析木のIDをそれぞれ5及び6とする。

【0184】

【数36】

【数37】

【0187】■処理C： 命令の合成

原料文書のGETと雛型文書のPUTのすべての組み合わせのうち、ラベル文字列が一致するものでペアを構成し、これらのペアから新しい命令を合成する。この動作手順について以下に説明する。

【0188】本実施例では、例えば”GET LABEL タグ名1 文書部品1”と”PUT LABEL タグ名2 属性名 文書部品2”というペアから”SUBST タグ名1 文書部品1 タグ名2 文書部品2 属性名”を合成する。SUBSTは、文書部品1に含まれるタグ名1のノードの属性フィールドから、属性名で指定した属性値を選択して、文書部品2に含まれるタグ名2のノードの属性フィールドに設定する命令である。この動作によって、雛型文書に含まれるタグの属性値を別のものに置き換えることが可能になる。属性値の置き換えは、タグ名が異なるペアであっても同じ名前の属性フィールドがあれば行われる。

【0189】

【数38】Step41： 原料文書のすべての命令列を走査し、ラベル文字列ごとに命令の配列を作成する。この配列をGET配列と呼ぶ。GET配列は、ラベル文字列をキーとする連想配列に格納しておく。

Step42： 雛型文書のすべての命令列を走査し、ラベル文字列ごとに命令の配列を作成する。この配列はPUT配列と呼ぶ。PUT配列は、ラベル文字列を属性として保持する。

Step43：すべてのPUT配列についてStep44以降を実行する。

Step44：PUT配列の属性であるラベル文字列と一致する文字列をオペランドに持つGET配列を取り出す。

Step45：PUT配列のすべての要素に対して、0番目から順にSTEP6以降を実行する

Step46：GET配列のm番目の要素として格納されたGET命令オペランド中のタグ名であるタグ名1と、文書部品IDである文書部品1を取り出す。PUT配列のm番目の要素として格納されたPUT命令オペランド中のタグ名であるタグ名2と属性名と文書部品IDである文書部品2を取り出す。

Step47： 文書処理記述に次の命令を追加する。この例では、以下に示すように、オペコードはSUBSTであり、また、オペランドはタグ名1、文書部品1、タグ名2、文書部品2、及び属性名である。

SUBST タグ名1 文書部品1 タグ名2 文書部品2 属性名

【0190】上記の処理によって、第3の原料文書と第3の雛型文書とから合成される文書処理記述の例を以下に示しておく。

【0191】

【数39】

SUBST A 5, [0 1 3] AREA 6, [0 1 3 1] HREF

【0192】この命令は、文書5（図11を参照のこと）のノード[0 1 3]以下のAタグのHREF属性の値を文書6（図12を参照のこと）のノード[0 1 31]以下のAREAタグのHREF属性に設定することを指示するものである。

【0193】図11及び図12に示した例では、原料文書中の文書部品にGET命令が付与されている場合を対象としたが、本実施例に係る構造化文書処理システムは、GET命令が雛型文書中の文書部品に付与されている場合であっても同様の処理を実現することができる。その場合、前述した命令の合成処理のStep42において、GET命令の配列とPUT命令の配列を別個に作成して、GET命令の配列をStep41で作成した配列に追加すればよい。

【0194】また、上述では置換する属性名がただ1つの場合について説明したが、属性置換開始命令中に属性名を”，”記号で区切って複数記述することによって、複数の属性値を一度に置き換えるようにしてもよい。

【0195】次いで、インタプリタと属性抽出手段、及び属性置換手段の動作について説明する。

【0196】インタプリタの基本動作は第1の実施例のそれと略同一であるので、相違する点を中心に説明する。インタプリタは、入力された文書処理記述のオペコードがSUBSTであれば、オペランドで指定された文書部品1の文書部品IDとタグ名1を属性抽出手段へ入力する。また、文書部品2の文書部品IDとタグ名2、及び属性名を属性置換手段へ入力する。

【0197】属性抽出手段は、インタプリタから入力された文書部品IDから、指定された文書解析木のノードをアドレスに従って取り出す。そして、そのノード以下から、入力されたタグ名に一致する対象タグ1のノードを行きがけ順に探索して、最初に見つかったノードを返す。

【0198】属性置換手段は、インタプリタから入力されたの文書部品IDから、指定された文書解析木のノードをアドレスに従って取り出す。そして、そのノード以下のすべてのノードの中から、入力されたタグ名に一致する対象タグ2のノードを行きがけ順に探索する。そして、見つかった対象タグ2のノードの属性フィールドの中で、インタープリタから入力された属性名と一致するものの属性値を、対象タグ1の属性名のものの属性値で置き換える。対象タグ2が複数ある場合は、この置き換える処理を繰り返す。

【0199】また、前記Step46とStep47に代えて、以下のStep46aとStep47aとしてもよい。

【0200】Step46a：GET配列のm番目の要素として格納されたGET命令オペランド中のタグ名であるタグ名1と、文書部品IDである文書部品1を取り出す。文書部品1のノードから、解析木を行きがけ順に

巡回して、タグ名１の対象タグ１を検索する。ＰＵＴ配列のｍ番目の要素として格納されたＰＩＴ命令オペランド中のタグ名であるタグ名２と属性名と文書部品ＩＤである文書部品２を取り出す。文書部品２のノードから、解析木を行きがけ順に巡回して、タグ名２の対象タグ２を検索する。

Ｓｔｅｐ４７ａ：　文書処理記述に次の命令を追加する。この例では、オペコードはＳＵＢＳＴで、オペランド

SUBST　5,［0　1　3］　6,［0　1　3　1］　HREF

【０２０３】この命令は、文書５（図１１を参照のこと）のノード［０　１　２］の（Ａタグの）ＨＲＥＦ属性値を文書６（図１２を参照のこと）のノード［０　１　３１］の（ＡＲＥＡタグの）ＨＲＥＦ属性に設定するものである。

【０２０４】インタプリタは、入力された文書処理記述のオペコードＳＵＢＳＴのオペランドで指定された対象タグ１の文書部品ＩＤと属性名を属性抽出手段へ入力する。また対象タグ２の文書部品ＩＤと属性名を属性置換手段へ入力する。

【０２０５】属性抽出手段と属性置換手段は、入力された文書部品ＩＤからアドレスに従って対象タグ１又は対象タグ２を取り出した後、対象タグ１の指定された属性名の属性値を取り出し、または対象タグ２の属性値と置換する。

【０２０６】第３の原料文書と第３の雛型文書とから合成した結果のＨＴＭＬファイルを以下に示す。

【０２０７】

【数４１】

```
<HTML>
<HEAD>
<TITLE> Template3 </TITLE>
</HEAD>
<BODY>
   <H1>資料集</H1>
   <HR>
   <IMG SRC="images/chart.gif" alt="チャート" WIDTH="600" HEIGHT="200" BORDER="0" USEMAP="#mapdata0">
   <MAP NAME="mapdata0">
<!--#PUT リンク AREA=HREF-->
   <AREA SHAPE="rect" ALT="リンク集" coords=0,20,200,85 HREF="http://sample.com/">
<!--#/PUT-->
</MAP>
<HR>
</BODY>
</HTML>
```

【０２０８】また、第３の原料文書と第３の雛型文書とから合成した結果のＨＴＭＬファイルを標準的なＨＴＭＬブラウザで表示した様子を図２３に示しておく。＃ｄ第３の雛型文書の例では、表示される画像（チャート：ｉｍａｇｅｓ／ｃｈａｒｔ．ｇｉｆ）の指定された座標位置（ｃｏｏｒｄｓ＝０，２０，２００，８５）をクリックすることにより、リンク先のページをたどっていくことができる。本実施例の文書処理によって、第３の雛型文書では指定されていなかったリンク先が第３の原料文書で指定されていたもの（ｈｔｔｐ：／／ｓａｍｐｌｅ．ｃｏｍ）に置換されている。

【０２０９】次いで、本発明の第３の実施例について説明する。本実施例では、抽出命令に代えてパス名付き抽出命令が、反復複写命令に代えてパターン式付き反復複写命令が、挿入置換命令に代えてパターン式付き挿入置換命令が、それぞれ付与された構造化文書に対して文書処理を実行するものである。以下、第１の実施例との相違点を中心に本実施例に関して説明する。

【０２１０】命令付与手段は、第１の実施例と同様に、ＨＴＭＬエディタによって構成される。本実施例に係る命令付与手段は、入力される文書内に、抽出命令に代えてパス名付き抽出命令を、反復複写命令に代えてパター

ン式付き反復複写命令を、挿入置換命令に代えてパターン式付き挿入置換命令を、それぞれ埋め込む。各命令は、次のように記述される。すなわち、

【0211】

【数42】・パス名付き抽出命令

<!--#OUT　パス名-->

※HTMLで記述された文書部品

<!--#/OUT-->

【0212】

【数43】・パターン式付き反復複写命令

<!--#RP　パターン式-->

※HTMLで記述された文書部品

<!--#/RP-->

【0213】

【数44】・パターン式付き挿入置換命令

<!--#IN　パターン式-->

※HTMLで記述された文書部品

<!--#/IN-->

【0214】入力手段、解析手段、及び命令分離分離手段の各々は第1の実施例と略同一である。但し、命令分離手段については、ラベル文字列の代りにパス名またはパターン式をオペランドとして抽出する。これは、上記第1の実施例において説明した正規表現による文字列マッチを用いて容易に実現可能であるので、ここでは詳しく説明しない。

【0215】本実施例では、第1の原料文書(図3を参照のこと)のラベル文字列の「グループ」及び「サマリ」に代えて、パス名として「月報/グループA」及び「月報/サマリA」にした構造化文書を第4の原料文書(原料文書4)として用いる。また、第2の原料文書(図4を参照のこと)のラベル文字列「グループ」及び「サマリ」に代えて、パス名「月報/グループB」及び「月報/サマリB」にした構造化文書を第5の原料文書(原料文書5)とする。また、第2の原料文書のラベル文字列「グループ」及び「サマリ」に代えて、パス名「月報/グループE」及び「月報/サマリE」にした構造化文書を第6の原料文書(原料文書6)とする。

【0216】本実施例では、パターン式は正規表現とする。

【0217】また、第1の雛型文書のラベル文字列の「グループ」及び「サマリ」に代えて、パターン式として「.*/グループ[A-D]」「.*/サマリ[A-D]」にした構造化文書を第4の雛型文書(雛型文書4)とする。ここで言う正規表現「.*/グループ[A-D]」の意味は、任意の文字列(.*)の後に'/グループ'が続き、次に、'A'か'B'か'C'か'D'のいずれかが続く文字列とマッチするパターンである。

【0218】命令分離手段によってそれぞれの文書から分離した命令列を以下に示す。

【0219】

【数45】

第4の原料文書から分離した命令列

オペコード	ラベル	文書部品ID
OUT	月報/グループA	<7, [0 1 0 1]>
OUT	月報/サマリA	<7, [0 1 4]>

【数46】

第5の原料文書から分離した命令列

オペコード	パス名	文書部品ID
OUT	月報/グループB	<8, [0 1 0 2]>
OUT	月報/サマリB	<8, [0 1 4]>

【数47】

第6の原料文書から分離した命令列

オペコード	パス名	文書部品ID
OUT	月報/グループE	<9, [0 1 0 2]>
OUT	月報/サマリE	<9, [0 1 4]>

【数48】

第4の雛型文書から分離した命令列

オペコード	パターン式	文書部品ID
RP	.*/グループ[A-D]	<10, [0 1 1 2]>
IN	.*/グループ[A-D]	<10, [0 1 1 2 0 1]>
IN	.*/サマリ[A-D]	<10, [0 1 1 2 1 1]>

【0223】次いで、本実施例の文書処理記述合成手段について説明する。但し、本実施例では、ラベル文字列

の一致に代えて、パス名とパターン式のパターン・マッチを行なう以外は、第１の実施例における文書処理記述合成手段と略同一の処理を行なう。ここでは、原料文書４と原料文書５と雛型文書４のそれぞれから分離した命令列から文書処理記述を構成する処理を例にとってその動作特性について説明することにする。

【０２２４】■処理Ａ：　ＯＵＴの計数

第１の実施例の処理Ａで行なったＯＵＴの計数は行なわない。

【０２２５】■処理Ｂ：　ＲＰによる各命令の複写

雛型文書１の命令列において、命令列中にＲＰが含まれていた場合、すべてのＲＰのオペランドの文書部品中に含まれるＩＮとＯＵＴとＲＰを、原料文書のＯＵＴ命令のうち、パス名とパターン式のパターン・マッチングを行なって、マッチが成功した個数に従って複写する。この複写は各１つの雛型文書について、次のＳｔｅｐ５１～Ｓｔｅｐ５６を実行することで行われる。正規表現のパターン・マッチングは、例えば、ＵＮＩＸ上のｇｒｅｐコマンド（周知）などを用いることで実現される。

【０２２６】

オペコード	パス名／パターン式	文書部品ＩＤ
ＲＰ	２	＜１０，［０　１　１　２］＞
ＩＮ	．＊／グループ［Ａ－Ｄ］	＜１０，［０　１　１　２　０　１］＞
ＩＮ	．＊／サマリ［Ａ－Ｄ］	＜１０，［０　１　１　２　１　１］＞
ＩＮ	．＊／グループ［Ａ－Ｄ］	＜１０，［０　１　１　３　０　１］＞
ＩＮ	．＊／サマリ［Ａ－Ｄ］	＜１０，［０　１　１　３　１　１］＞

【０２２９】■処理Ｃ：　命令の合成

原料文書中のＯＵＴ命令と雛型文書中のＩＮ命令とのすべての組み合わせのうち、パス名とパターン式のパターン・マッチングが成功するものどうしでペアを構成し、これらのペアから新しい命令を合成する。以下、この命令の合成動作について説明する。

【０２３０】本実施例では、例えば”ＯＵＴ　パス名　文書部品１”と”ＩＮ　パターン式文書部品２”というペアから”ＦＩＬＬ　文書部品１　文書部品２”を合成する。ＦＩＬＬは、文書部品２を文書部品２と置換する命令を表す。この動作によって、原料文書に出現する文書部品の出現順にかかわらず、雛型文書の中でユーザが所望する任意の位置に、原料文書の文書部品を置換又は挿入することが可能になる。

【０２３１】

【数５１】Ｓｔｅｐ６１：　原料文書のすべての命令列を走査して、ＯＵＴ命令を配列に格納する。この配列はＯＵＴ配列と呼ぶ。ＯＵＴ配列は、パス名を属性として保持する。

Ｓｔｅｐ６２：雛型文書のすべての命令列を走査し、ＩＮ命令を配列に格納する。これをＩＮ配列と呼ぶ。ＩＮ配列は、パターン式を属性として保持する。

【数４９】Ｓｔｅｐ５１：ＲＰリストを作成する。

Ｓｔｅｐ５２：ＲＰリストから、対象とするＲＰ（反復複写命令）を順に変更しつつ各ＲＰについてＳｔｅｐ５３以降を実行する。

Ｓｔｅｐ５３：対象Ｒｐのオペランドのパターン式を取り出し、命令列を先頭から順に走査してＯＵＴのオペランドのパス名を取り出し、パターン・マッチングを行なって、成功したＯＵＴ命令の個数を反復複写回数ｔとする。ｔが２以上ならＳｔｅｐ５４以降を実行する。Ｓｔｅｐ５４以降は、第１の実施例におけるＳｔｅｐ２４以降と略同一なので、ここでは説明を省略する。

【０２２７】パターン式「．＊／グループ［Ａ－Ｄ］」は、「月報／グループＡ」と「月報／グループＢ」とのパターン・マッチは成功するが、「月報／グループＥ」とのパターンマッチは失敗する。したがって、上述の例では複写回数ｔ＝２となる。

【０２２８】

【数５０】第４の雛型文書から分離した命令列に対して上記の処理Ｂを施した結果得られる命令列

Ｓｔｅｐ６３：ＩＮ配列のすべてのＩＮ命令について、Ｓｔｅｐ６４以降を順次実行する。

Ｓｔｅｐ６４：ＩＮ配列の属性であるパターン式を取り出し、ＯＵＴ配列の要素を先頭から順に走査して、要素のパス名とパターン・マッチングを行ない、成功した要素からＯＵＴ命令を取り出す。成功したＯＵＴ配列の要素は取り除く。

Ｓｔｅｐ６５：ＯＵＴ配列から取り出したＯＵＴ命令のオペランドの文書部品のＩＤ　である文書部品１を取り出す。また、ＩＮ命令のオペランドの文書部品のＩＤである文書部品２を取り出す。

Ｓｔｅｐ６６：　文書処理記述に次の命令を追加する。この例では、命令のオペコードはＦＩＬＬで、オペランドは文書部品１及び文書部品２である

ＦＩＬＬ　　文書部品１　　文書部品２

【０２３２】上記の処理手順によって文書処理記述が合成される。合成された文書処理記述の例を以下に示す。

【０２３３】

【数52】

※第1の原料文書と第2の原料文書と第1の雛型文書から合成された文書処理記述例1

```
RP      2          2,[0 1 1 2]
FILL    0,[0 1 0 1]    2,[0 1 1 2 0 1]
FILL    1,[0 1 0 2]    2,[0 1 1 3 0 1]
FILL    0,[0 1 4]      2,[0 1 1 2 1 1]
FILL    1,[0 1 4]      2,[0 1 1 3 1 1]
```

【0234】上述した例では、パターン・マッチに失敗したOUT命令のオペランドで指定された文書部品は、雛型文書に挿入又は置換が行われないことに留意されたい。

【0235】本実施例に係る構造化文書処理システムによって、原料文書4と原料文書5と原料文書6と雛型文書4から合成されるHTMLファイルを以下に示す。

【0236】

【数53】

```
<HTML>
<HEAD>
<TITLE> Template1 </TITLE>
</HEAD>
<BODY BGCOLOR="#404040">
   <H1>進捗サマリ</H1>
   <TABLE BORDER="1">
          <TR><TH>チーム</TH><TH>概況</TH><
/TR>
<!--#RP .*/グループ[A-D]-->
      <TR>
          <TH>
<!--#IN .*/グループ[A-D]-->Aチーム<!--#/IN-
->
          </TH>
          <TD>
<!--#IN .*/サマリ[A-D]--><UL>
               <LI> 項目1の進捗60% 計画どおり </L
I>
               <LI> 項目2の進捗30% 遅れ: 3日 </
LI>
               <LI> 項目3の進捗70% 予定外の問題発生、
計画見直しが必要 </LI>
          </UL><!--#/IN-->
          </TD>
      </TR>
      <TR>
          <TH>
<!--#IN .*/グループ[A-D]-->Bチーム<!--#/IN-
->
          </TH>
          <TD>
<!--#IN .*/サマリ[A-D]--><TABLE BORDER=
"0" CELLSPACING="0" CELLPADDING="1">
               <TR><TH>項目</TH><TH>進捗</T
H><TH ALIGN="LEFT"">評価</TH></TR>
               <TR><TH>1</TH><TD>20%</T
```

```
D><TD>やや計画おくれ（1日）</TD></TR>
                    <TR><TH>2</TH><TD>40%</T
D><TD>遅れなし</TD></TR>
                    <TR><TH>3</TH><TD>60%</T
D><TD>進み：　3日</TD></TR>
              </TABLE><!--#/IN-->
              </TD>
        </TR>
<!--#/RP-->
</TABLE>
</BODY>
</HTML>
```

【0237】図24には、上記のHTMLファイルを標準的なHTMLブラウザ上で表示した様子を示している。

【0238】本発明の第3の実施例は、第1の実施例と同様の作用効果を奏することができる。但し、ラベル文字列に代えてパターン式を用いたことにより、より柔軟で複雑な文書処理記述を合成することが可能となる点には充分理解されたい。

【0239】次いで、本発明の第4の実施例について説明する。本実施例は、文書処理記述合成手段と挿入置換手段に代えて、以下に説明する文書処理記述合成手段と挿入置換手段を有するという点で、上記の第3の実施例とは顕著に相違する。

【0240】本実施例における命令文書処理記述合成手段が実行する処理Cについて以下に説明する。

【0241】■処理C：　命令の合成

原料文書中のOUT命令と雛型文書中のIN命令とのすべての組み合わせのうち、パス名とパターン式のパターン・マッチングが成功するものどうしでペアを構成し、これらペアから新しい命令を合成する。以下、この命令の合成動作について説明する。

【0242】本実施例では、例えば”OUT　パス名　文書部品1”と”IN　パターン式文書部品2”というペアから”FILL　文書部品1　文書部品2　パス名”を合成する。FILLは、文書部品2を文書部品2と置換する命令を表す。FILLは、文書部品1を文書部品2と置換する命令を表す。加えて、挿入置換命令開始タグと挿入置換命令終了タグを、抽出命令開始タグと抽出命令終了タグにそれぞれ置換する命令を表す。この動作によって原料文書に出現する文書部品の出現順にかかわらず、雛型文書中でユーザが所望する任意の位置に、原料文書の文書部品を置換又は挿入することが可能になる。加えて、処理をした結果合成された文書を入力とし、本実施例に係る構造化文書処理システムによって再び処理する際には、元の原料文書に含まれる文書部品を識別することが可能になる。

【0243】

【数54】Step71：原料文書のすべての命令列を走査し、OUT命令を配列に格納する。この配列をOUT配列と呼ぶ。OUT配列は、パス名を属性として保持する。

Step72：雛型文書のすべての命令列を走査し、IN命令を配列に格納する。これをIN配列と呼ぶ。IN配列は、パターン式を属性として保持する。

Step73：IN配列中のすべてのIN命令について、Step74以降を順次実行する。

Step74：IN配列の属性であるパターン式を取り出し、OUT配列の要素を先頭から順に走査して、要素のパス名とパターン・マッチングを行ない、成功した要素からOUT命令を取り出す。成功したOUT配列の要素は取り除く

Step75：OUT配列から取り出したOUT命令のオペランドの文書部品のIDである文書部品1を取り出す。加えて、OUT命令のパス名を取り出すとともに、IN命令のオペランドの文書部品のINである文書部品2を取り出す。

Step76：　文書処理記述に次の命令を追加する。この例では、オペコードはFILLで、オペランドは文書部品1と文書部品2とパス名である

FILL　　文書部品1　　文書部品2　　パス名

【0244】パス名は、原料文書から分離した抽出命令（IN命令）のオペランドから取り出したパス名文字列に加えて、所定の名前の環境変数やグローバル変数に代入されている文字列、あるいは、本実施例に係る構造化文書処理システムを起動した際に引数として引き渡される文字列を、適当な区切り文字で区切って付与してもよい。付与する前記文字列を履歴文字列と呼ぶ。上述した例では、区切り文字として”／”を用い、履歴文字列としてホスト名を与えている。但し、ホスト名は”総務部”とする。この処理によって合成された文書処理記述の例を示す。

【0245】

【数55】

※第4の原料文書と第5の原料文書と第4の雛型文書から合成された文書処理記

述例
RP　　2　　2,[0 1 1 2]
FILL　0,[0 1 0 1]　2,[0 1 1 2 0 1]
総務部／月報／グループA
FILL　1,[0 1 0 2]　2,[0 1 1 3 0 1]
総務部／月報／グループB
FILL　0,[0 1 4]　2,[0 1 1 2 1 1]　総務部／月報／サマリA
FILL　1,[0 1 4]　2,[0 1 1 3 1 1]　総務部／月報／サマリB

【0246】次いで、インタプリタと挿入置換手段について説明する。

【0247】インタプリタは、文書処理記述のFILL命令を解釈して、挿入置換手段に、文書部品1と文書部品2とパス名を入力する。これ以外の点については実施例1のインタプリタと略同一である。

【0248】挿入置換手段は、文書部品2で指定されるIDの文書部品にもともと付与されている挿入置換命令である挿入置換命令開始タグと挿入置換命令終了タグを、それぞれ抽出命令開始タグと抽出命令終了タグに置き換える。また、インタプリタから入力されたパス名を抽出命令開始タグに付与する。この置き換え処理は、HTMLファイル中のそれぞれの注釈タグの注釈文字列を置き換えることによって容易に実現できる。この点以外は、上述した第1の実施例における挿入置換手段と略同一である。

【0249】本実施例に係る構造化文書処理システムによって、原料文書4と原料文書5と雛型文書4から合成されるHTMLファイルを以下に示す。

【0250】

【数56】

```
<HTML>
<HEAD>
<TITLE> Template1 </TITLE>
</HEAD>
<BODY BGCOLOR="#404040">
   <H1>進捗サマリ</H1>
   <TABLE BORDER="1">
          <TR><TH>チーム</TH><TH>概況</TH><
/TR>
<!--#RP .*/グループ[A−D]-->
      <TR>
          <TH>
<!--#OUT 総務部／月報／グループA−−>Aチーム<!−−#／OU
T−−>
          </TH>
          <TD>
<!−−#OUT 総務部／月報／サマリA−−><UL>
                <LI> 項目1の進捗60% 計画どおり </L
I>
                <LI> 項目2の進捗30% 遅れ： 3日 </
LI>
                <LI> 項目3の進捗70% 予定外の問題発生、
計画見直しが必要 </LI>
          </UL><!−−#／OUT−−>
          </TD>
      </TR>
      <TR>
          <TH>
<!−−#OUT 総務部／月報／グループB−−>Bチーム<!−−#／OU
```

```
T-->
            </TH>
            <TD>
<!--#OUT 総務部/月報/サマリB--><TABLE BORDER
="0" CELLSPACING="0" CELLPADDING="1"
>
                <TR><TH>項目</TH><TH>進捗</T
H><TH ALIGN="LEFT">評価</TH></TR>
                <TR><TH>1</TH><TD>20%</T
D><TD>やや計画おくれ(1日)</TD></TR>
                <TR><TH>2</TH><TD>40%</T
D><TD>遅れなし</TD></TR>
                <TR><TH>3</TH><TD>60%</T
D><TD>進み: 3日</TD></TR>
            </TABLE><!--#/OUT-->
            </TD>
        </TR>
<!--#/RP-->
</TABLE>
</BODY>
</HTML>
```

【0251】図25には、上記のHTMLファイルを標準的なHTMLブラウザ上で表示した様子を示している。

【0252】本発明の第4の実施例は、第1の実施例と同様の作用効果を奏することができるとともに、さらに、原料文書の抽出命令とパス名を合成結果の構造化文書の文書部品に継承して付与することができる。

【0253】本実施例に係る構造化文書処理システムを複数段接続してパイプラインを構成することによって、最初の構造化文書処理システムで出力された原料文書に含まれている文書部品は、2段目以降の構造化文書処理ではパス名を調べることで識別することができる。この機能によって、より柔軟で複雑な構造化文書処理を実現することが可能となる。

【0254】次いで、本発明の第5の実施例について説明する。

【0255】上述した第1の実施例では、構造化文書処理システムは、HTTPリクエストを入力して、所定の構造化文書の処理を行ない、合成した構造化文書をHTTPレスポンスとして出力するHTTPサーバとして構成される。これに対し、本実施例では、HTTPリクエストに処理起動記述が埋め込まれている。ここで言う処理起動記述は、原料文書のURLと雛型文書のURLの双方を含むものとする。

【0256】本実施例に係る構造化文書処理システムは、このような処理起動記述を解析して、原料文書のURLと雛型文書のURLとを取り出す処理起動記述解析手段をさらに具備している。そして、処理起動記述解析手段によって解析した結果として取り出されたURLに基づいて、入力手段により原料文書や雛型文書を入力して処理することができる。図13には、本実施例に係る構造化文書処理システムの構成を模式的に示している。以下、第1の実施例と相違する点を中心に該システムの構成及び動作特性について説明する。

【0257】処理起動記述は、HTTPリクエストに、構造化文書処理装置の原料文書のURLと、雛型文書のURLを埋め込むことで構成される。本実施例では、特に、HTTPリクエストに含まれているURL文字列中に、所定の文法形式で、原料文書のURLと雛型文書のURLが埋め込まれる。

【0258】HTTPリクエストのURL中に別のURLを埋め込んだものを「ヴァーチャルURL」と呼ぶ。すなわち、本実施例に係る処理起動記述はヴァーチャルURLとして構成される。なお、処理起動記述の文法と、該文法に従って埋め込む方法と取り出す方法に関しては、例えば、本出願人に既に譲渡されている特許第2746218号公報(特開平8-292910号)「資源管理装置および資源管理方法」に詳述されているので、必要であれば参照されたい。

【0259】処理起動記述解析手段は、HTTPリクエストを所定の文法形式に従って解析して、原料文書のURLと雛型文書のURLの各々を取り出す。構造化文書処理システムにおける処理起動記述解析手段以外の機能ブロックの構成及び動作特性は、第1の実施例と略同一である。

【0260】構造化文書処理サーバは、HTTPリクエストのURL中に原料文書のURLと雛型文書のURLを含んだ形式の処理起動記述を受け取って、この処理起

動記述に指定された原料文書と雛型文書を入力して文書処理記述を合成して、さらに該文書処理記述を解釈して構造化文書処理を行なうことができる。

【0261】本実施例において利用される処理起動記述の例を以下に示す。

【0262】

【数57】原料文書1のURL： http://host1/Sample1.html
原料文書2のURL： http://host2/Sample2.html
雛型文書1のURL： http://host3/Template1.html
構造化文書処理サーバのURL： http://server/HTMLTools?method=fill
処理起動記述（HTTPリクエストのURL）：
http://server/HTMLTools?method=fill&template=http%3A%2F%2Fhost3%2FTemplate1.html&file1=http%3A%2F%2Fhost1%2FSample1.html&file2=http%3A%2F%2Fhost2%2FSample2.html

【0263】上記に示した処理起動記述の例では、雛型文書のURLは、文字列"template="以降で、次の"&"の前までに埋め込まれている。元々のURLに含まれていた":"と"/"は、それぞれ"%3A"と"%2F"にエンドコードされ変換されている。また、原料文書1と原料文書2に関するURLについても同様に、それぞれ文字列"file1="以降から次の"&"の前まで、並びに、文字列"file2="以降に埋め込まれている。

【0264】特許第2746218号公報に示すようなURLの表記方式によれば、2段階に構造化文書処理サーバを起動するヴァーチャルURLを階層的に構成することができる。例えば、原料文書や雛型文書は既に他の構造化文書処理サーバで処理した結果であってもよい。但し、この場合の原料文書や雛型文書のURLは、第2の構造化文書処理サーバのURLに基づくヴァーチャルURLである。

【0265】

【数58】原料文書3のURL： 前記HTTPリクエストのURL（処理結果のヴァーチャルURL）
雛型文書2のURL： URL： http://host3/Template2.html
第2の構造化文書処理サーバのURL： http://server2/HTMLTools?method=fill
前記文書処理記述を原料文書とする第2の処理起動記述（第2のHTTPリクエストのURL）：
http://server2/HTMLTools?method=fill&template=http%3A%2F%2Fhost3%2FTemplate2.html&file1=http%3A%2F%2Fserver%2FHTMLTools%3Fmethod%3Dfill%3Ftemplate%3Dhttp%253A%252F%252Fhost3%252FTemplate1.html%26file1%3Dhttp%253A%252F%252Fhost1%252FSample1.html%26file2%3Dhttp%253A%252F%252Fhost2%252FSample2.html

【0266】図14には、上述したような第2の処理起動記述によって起動される処理の流れを示している。

【0267】第2の構造化文書処理サーバ（server2）には、雛型文書2（Template2.html）を用いて原料文書3を処理する旨のHTTPリクエストが入力される。このHTTPリクエストはヴァーチャルURLを用いて記述される。

【0268】雛型文書2（Template2.html）は、WWWサーバ（host3）で提供されており、通常のURL表記" http://host3/Template2.html"によって指定される。

【0269】他方、原料文書3は、第1の構造化文書処理サーバ（server1）によって処理された結果である。より具体的には、該処理結果とは、WWWサーバ（host 1）の資源である原料文書1（Sample1.html）と、WWWサーバ（host 2）の資源である原料文書2（Sample2.html）とを、WWWサーバ（host3）の資源である雛型文書1（Template1.html）を用いて処理したものである。

【0270】したがって、第1の構造化文書処理サーバは、原料文書1、原料文書2、及び雛型文書1を各WWWサーバから入力して、その処理結果である原料文書3を第2の構造化文書処理サーバに渡す。

【0271】次いで、第2の構造化文書処理サーバは、さらにWWWサーバ（host3）から原料文書2を入力して、原料文書3を処理して、この処理結果である構造化文書をHTTPレスポンスとして要求元に返す。

【0272】図13に示したような構造化文書処理システムによれば、分散的に文書処理命令を管理し処理することができる。したがって、文書の改定や処理システムの変更に対して、より柔軟な文書処理を提供することができる。文書を処理するサーバをモジュールとして取り扱うことができるので、原料文書や雛形文書と文書処理モジュールを文書処理記述上で組み合わせることで、エンド・ユーザが容易にカストマイズすることができる柔軟な文書処理システムを構築することができる。

【0273】次いで、本発明の第6の実施例について説

明する。
【０２７４】本実施例は、上記第５の実施例に係る第１の構造化文書処理サーバ及び第２の構造化文書処理サーバが、単一のデータベースを共有する構成を備えているものである。
【０２７５】データベースには、識別子としてＵＲＬ又は処理起動記述をキーとして、このＵＲＬ又は処理起動記述に対応する構造化文書の解析木が格納されている。したがって、ＵＲＬ又は処理起動記述を指定することで、所望の構文解析木を取り出すことができる。このようなデータベースは、ＵＲＬ又は処理起動記述から適当なハッシュ値を計算し、解析木を実現するデータへのポインタを格納したハッシュ法による表の探索で実現できる（ハッシュ法については、例えば「岩波書店アルゴリズムとデータ構造」２．７節　ｐ１２３－１３７に詳述されている）。
【０２７６】図１５には、本実施例に係る第１の合成サーバの機能ブロック図を示している。本実施例においては、第２の（すなわち後続の）合成サーバは、第１の合成サーバと略同一の構成及び動作特性を有するものと理解されたい。
【０２７７】第１の合成サーバは、ＨＴＴＰリクエストを入力して、所定の構造化文書の処理を行ない、合成した構造化文書をＨＴＴＰレスポンスとして出力するＨＴＴＰサーバとして実現されている。本実施例は、以下の２点を除き、第１の実施例と略同一である。
【０２７８】（１）抽出手段、反復複写手段、並びに置換挿入手段は、解析手段から構文解析木を入力する代わりに、切り替え手段を経由して構文解析木を入力する。（２）切り替え手段は、システムの状態を調べて、解析手段から構文解析木を入力するか、又は、保持手段から構文解析木を入力するかを切り替える。
【０２７９】システムの状態は、例えば、処理中の処理起動記述をキーとして、保持手段の中に該当する構文解析木があるかどうかを調べることで判定できる。
【０２８０】保持手段に該当する構文解析木が格納されていれば、改めて構造文書を解析する必要はないので、保持手段から構文解析木を入力するように切り替える。この場合、構文解析木は、図１５中の記号（い）で示された一点破線の矢印方向に従って処理される。
【０２８１】他方、保持手段に該当する構文解析木が格納されていない場合には、切り替え手段は、解析手段から新たに構文解析木を入力するように切り替える。この場合、図１５中のの記号（ろ）で示された一点破線の矢印方向に従って、入出力が行われる。
【０２８２】本実施例に係る構造化文書処理システムの構成によれば、第１の合成サーバで処理した結果は、構文解析木のまま第２（後続）の合成サーバに入力することができる。したがって、出力手段による構造化文書のＨＴＭＬへの変換と、解析手段によるＨＴＭＬの解析という２つの処理を省略することができる。
【０２８３】また、第１の合成サーバにおいて一旦計算した結果は保持手段に格納されるので、第２の合成サーバが第１の合成サーバに処理起動記述を入力して第１の合成サーバが改めて処理を行う代わりに、既に計算されている前記処理起動記述に対応する構文解析木を利用することで、第１の合成サーバでの計算や合成処理を省略することができる。
【０２８４】したがって、従来に比べ処理性能の優れた効率的な分散文書処理を実現することができる。
【０２８５】次いで、本発明の第７の実施例について説明する。
【０２８６】図１６には、本発明の第７の実施例に係る構造化文書処理システムの構成を模式的に示している。本実施例は、上述した第２の実施例の構成に対して、第５の実施例に係る処理起動記述解析手段をさらに追加したものである。
【０２８７】第２の実施例では、属性抽出命令及び属性置換命令がそれぞれ指定していた文書部品の間で属性置換を行っていた。これに対し、本実施例では、属性抽出命令が文書部品ではなく手続き名を指定する。属性置換命令は、指定した文書部品の属性を、属性抽出命令で指定された手続きで変換したものに置き換える。処理起動記述解析手段については第５の実施例のそれと略同一であるので、以下では第２の実施例との相違点を中心に説明することにする。
【０２８８】本実施例では、属性抽出命令を以下のような形式で記述する。
【0.289】
【数５９】・属性抽出命令
<！－－＃ＧＥＴ　ＬＡＢＥＬ　手続き名－－>
<！－－＃／ＧＥＴ－－>
【０２９０】第２の実施例の場合とは相違し、属性抽出命令には置換する対象のタグ名を指定する必要がなく、開始タグと終了タグの間に文書部品を挟む必要もない。属性置換命令は、第２の実施例と同一である。
【０２９１】属性抽出命令を付与したＨＴＭＬファイルを第４の原料文書とし、以下に示す。
【０２９２】
【数６０】<ＨＴＭＬ>
<ＨＥＡＤ>
<ＴＩＴＬＥ>　Ｓａｍｐｌｅ３　</ＴＩＴＬＥ>
</ＨＥＡＤ>
<ＢＯＤＹ>
<Ｈ２>リンク集</Ｈ２>
<ＨＲ>
<！－－＃ＧＥＴ　リンク　Ｐｒｏｃ１－－>
<！－－＃／ＧＥＴ－－>
<ＨＲ>
</ＢＯＤＹ>

</HTML>

【0293】解析手段は、注釈タグが以下の規則に従うかを解析する。

【0294】

【数61】"#GET<空白><LABEL文字列><空白><手続き名文字列>" →属性抽出命令開始タグ

"#/GET" → 属性抽出命令終了タグ

※属性置換命令については第2の実施例と同じ。

【0295】属性抽出命令には対象タグがないので、文書部品から対象タグを探す処理も行わない。

【0296】命令分離手段も第2の実施例と同様であるが、属性抽出命令にはタグ名ではなく手続き名が引数として付与されているため、これをオペランドとして出力する。命令分離手段によって第4の原料文書から分離した命令列を以下に示す。

【0297】

【数62】オペコード ラベル 手続き名

GET リンク Proc1

【0298】文書処理記述合成手段は、処理Cすなわち命令の合成において、第2の実施例と相違する。文書処理記述合成手段は、属性抽出命令と属性置換命令とのペアから新しい命令を合成するが、例えば、"GET LABEL 手続き名"と"PUT LABEL タグ名 属性名 文書部品"のペアから、"SUBST 手続き名 タグ名 文書部品 属性名"を合成する。SUBSTは、文書部品に含まれるタグ名のノードの属性フィールドから属性名で指定した属性値を選択して、手続き名で指定した手続きで属性値を変換する命令である。

SUBST Proc1 AREA 6,[0 1 3 1] HREF

【0299】処理手順は、Step46以降が第2の実施例とは相違し、以下の通りとなる。

【0300】

【数63】Step46b: GET配列のm番目の要素として格納されたGET命令オペランド中の手続き名を取り出す。PUT配列のm番目の要素として格納されたPUT命令オペランド中のタグ名と属性名、文書部品IDである文書部品を取り出す。

Step47b:文書処理記述に次の命令を追加する。この例では、オペコードはSUBSTで、オペランドは、手続き名、タグ名、文書部品、属性名である。SUBST 手続き名 タグ名 文書部品 属性名

【0301】上記の処理によって、第4の原料文書と第3の雛型文書とから合成される文書処理記述例を以下に示す。

【0302】

【数64】

【0303】この命令は、文書6のノード[0 1 3 1]以下のAREAタグのHREF属性の値をProc1で変換したものに置き換えることを指示するものである。

【0304】インタプリタは、SUBST命令を属性置換手段に入力する。本実施例の属性置換手段の動作は、第2の実施例のそれとは異なり、SUBST命令を解釈して次のように動作する。すなわち、対象タグのHREF属性の属性値を第1の引数として、手続きProc1を呼び出して、Proc1の返り値として得られた文字列を新たなHREF属性として対象タグに設定する。但し、手続きの呼び出しは普通に知られる手続き呼び出し(サブルーチン・コール)であり、HREF属性の取り出しやHREF属性の設定は通常の文字列の分離又は結合である。

【0305】属性値を変換する手続きとしては、例えば次のようなものが考えられる。属性置換命令を付与した文書部品が属性値フィールドにURLを設定できる種類のタグのノードを含んでいる場合には、そのURL文字列を第5の実施例で説明した処理起動記述(ヴァーチャルURL)に変換する。例えば、元のURLに設定されている雛型文書と文書を特定する文字列を解析し、それらを元に新たな処理起動記述を合成して返すような手続きである。このようなURL文字列は、手続きの引数として受け渡される。

雛型文書のURL: http://host3/Template2.html

変換前のURL(): http://host3/Template2.h

【0306】ここで、変換前のURLの記述例と変換後の処理起動記述の例を示す。但し、この例では、処理起動記述解析手段に入力された処理起動記述が含んでいる原料文書のうち2番目のものを、新たに合成する処理起動記述に含める原料文書とする。また、新たに合成する処理起動記述に含める雛型文書は、入力された処理起動記述が含んでいる雛型文書とは別のものにする。

【0307】

【数65】元の処理起動記述(処理起動記述解析手段に入力されたもの):

http://server/HTMLTools?method=fill&template=http%3A%2F%2Fhost3%2FTemplate1.html&file1=http%3A%2F%2Fhost1%2FSample1.html&file2=http%3A%2F%2Fhost2%2FSample2.html

【0308】

【数66】

tml?file2=

【0309】

【数67】変換後のURL(処理起動記述):

http://server/HTMLTools?method=fill&template=http%3A%2F%2Fhost3%2Ftemplate2.html&file2=http%3A%2F%2Fhost2%2FSample2.html

【0310】また、手続きの別の例として、新たに処理起動記述を合成するのではなく、元の処理起動記述に含まれる原料文書のURLをそのまま返すようなものも考えられる。

【0311】変換前のURLの記述例と変換後の処理起動記述の例を以下に示す。但し、処理起動記述解析手段に入力された処理起動記述が含んでいる原料文書のうちの最初のものを原料文書とする。

【0312】

【数68】元の処理起動記述(処理起動記述解析手段に入力されたもの):

http://server/HTMLTools?method=fill&template=http%3A%2F%2Fhost3%2FTemplate1.html&file1=http%3A%2F%2Fhost1%2FSample1.html&file2=http%3A%2F%2Fhost2%2FSample2.html

【0313】

【数69】

変換前のURL():　dummy.html#file1

【0314】

【数70】変換後のURL(処理起動記述):

http://host1/Sample1.html

【0315】上記の例で示した変換前のURLは、ここで定義した独自の記述方法である。変換前のURLは、第3の雛型文書の例ではAREAタグのHREF属性値として設定すればよい。

【0316】また、手続きの引数としてOSのシステム・コールを用いて得られる日時やシステムの状態変数などを利用するようにしてもよい。

【0317】次いで、本発明の第8の実施例について説明する。

【0318】図17には、本発明の第7の実施例に係る構造化文書処理システムの構成を模式的に示している。本実施例は、上述した第6の実施例の構成に対して、処理起動記述合成手段と、エラー通知文書合成手段をさらに追加したものである。

【0319】本実施例の命令分離手段は、第1の実施例に係る命令分離手段が有する動作に加え、命令列を抽出する際に検出される命令の文法的なエラーを検出して、エラー情報をエラー通知文書合成手段に入力するようになっている。ここで言うエラー情報は、エラーの種類やエラーの検出された文書中の位置(行数)など、通常のエラー処理でよく扱われるものとする(命令分離手段におけるエラーの検出やエラー処理の実現方法については、例えば、岩波講座ソフトウェア科学5 「プログラミング言語処理系」(岩波書店)の P89, P123-125, 206-208に詳解されている)。

【0320】また、本実施例の出力手段は、第1の実施例に係る出力手段が持つ機能に加えて、処理起動記述合成手段によって合成された処理起動記述を、出力結果としてのHTML形式文書に挿入するように動作するようになっている。この動作手順について以下に説明する。

【0321】エラー通知文書合成手段は、エラー情報を入力して、エラー通知を行なう文書を合成する。エラー情報は例えばエラー番号とし、エラー文書はエラー番号をファイル名とするファイルとする。エラー通知文書合成手段は、命令解析手段から入力されたエラー番号をファイル名とみなして、ファイルからエラー通知文書の1つを入力する。入力された文書は、解析手段を用いて構文解析木が作られる。構文解析木を作る動作は、第1の実施例の場合と同様である。構文解析木は後述するように保持手段に保持されるが、必ずしも構文解析木を作る必要はなく、文字列情報のまま保持手段に保持してもよい。例えば、エラー番号1は、抽出命令開始タグと抽出命令終了タグの対応がとれないエラーであるとし、エラー通知文書のファイル名はファイル1.htmlとする。ファイルの例を示す。

【0322】

【数71】<HTML>
<HEAD>
<TITLE> エラー1</TITLE>
<BODY>
原料ファイルの処理中に抽出命令開始タグと抽出命令終了タグの対応がとれないエラーが発生した。
</BODY>
</HTML>

【0323】処理起動記述合成手段は、エラー通知ページをアクセスするためのURLを合成する。例えば、この文書処理を行なった合成サーバが"server"であって、文書処理を行なったプロセスのプロセス番号が12345番であったとすると、以下に示すようにプロセス番号を含むURLを合成する。

【0324】

【数72】http://server/HTMLTools?method=userlog&pid=12345

【0325】次いで、本実施例の出力手段の動作につい

て、第１の実施例との相違点を中心に説明する。例えば、上述したような処理起動記述が入力されると、文字列<META USERLOG=" http://server/HTMLTools?method=userlog&pid=12345">を合成して、出力されるHTML形式の文書に挿入する。このような挿入は、通常知られる単純な文字列合成や文字列挿入処理である。挿入された結果のHTML形式の文書の一部（例７－Ａ）を以下に例示しておく。

【０３２６】

【数７３】<HTML>
<HEAD>
<META USERLOG=" http://server/HTMLTools?method=userlog&pid=12345">
</HEAD>
...

【０３２７】また、別の文字列を合成して挿入した例７－Ｂを以下に示す。

【０３２８】

【数７４】<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript"> <!--window.open("http://server/HTMLTools?method=userlog&pid=12345");
// -->
</SCRIPT>
</HEAD>

【０３２９】エラー通知文書合成手段で合成されたエラー通知文書は、処理起動記述合成手段で合成されたURLをキー情報として、保持手段に格納され保持される。この保持手段に文書を格納する動作は第６の実施例とは相違する。

【０３３０】上述した第６の実施例では、WWWクライアントからのリクエストすなわちHTTPリクエストとして、第１の処理起動記述が与えられる。次いで、処理起動記述処理が行なわれ、処理結果として合成された文書が第１の処理起動記述をキーとして保持手段に格納される。これに対し、本実施例では、後のエラー通知文書を得るための第２の処理起動記述を、エラー通知文書のリクエストに先行して合成サーバ側で合成し、（実際のリクエストが入力されるより前に）保持手段に格納するように動作するようになっている。

【０３３１】合成サーバserverにおいて文書処理を実行した際にエラーが発生した場合を例にとって以下に説明してみる。

【０３３２】上述した例７－Ａのタイプの場合には、利用者は、serverから出力されるHTTPレスポンスによって送信されるHTML形式の文書の内容からUSERLOG属性を有するMETAタグを検索して、エラー通知文書の処理起動記述に相当するURLを得る。利用者は、このURLを含む第２のHTTPリクエストをserverに送信する。合成サーバは、処理起動記述解析手段によりHTTPリクエストに含まれる処理起動記述を取り出し、これをキーとして保持手段から文書を検索し、エラー通知文書を得る。それ以降は、第１の実施例の場合と同様の動作手順を実行して、新たなHTTPレスポンスとしてエラー通知文書を出力する。

【０３３３】エラー通知文書を雛型文書として取り扱えば、適当な文書処理が行われて、出力される。この場合、エラーが発生した位置（行数）やエラーが発生した文書のＩＤなどを挿入することが考えられる。このような文書処理に対する引数（入力パラメータ）は、ＣＧＩ（Common Gateway Interface）として当業界において広く知られているHTTPサーバに対するURLの記述方法に従って文書処理記述中に含めることができる。

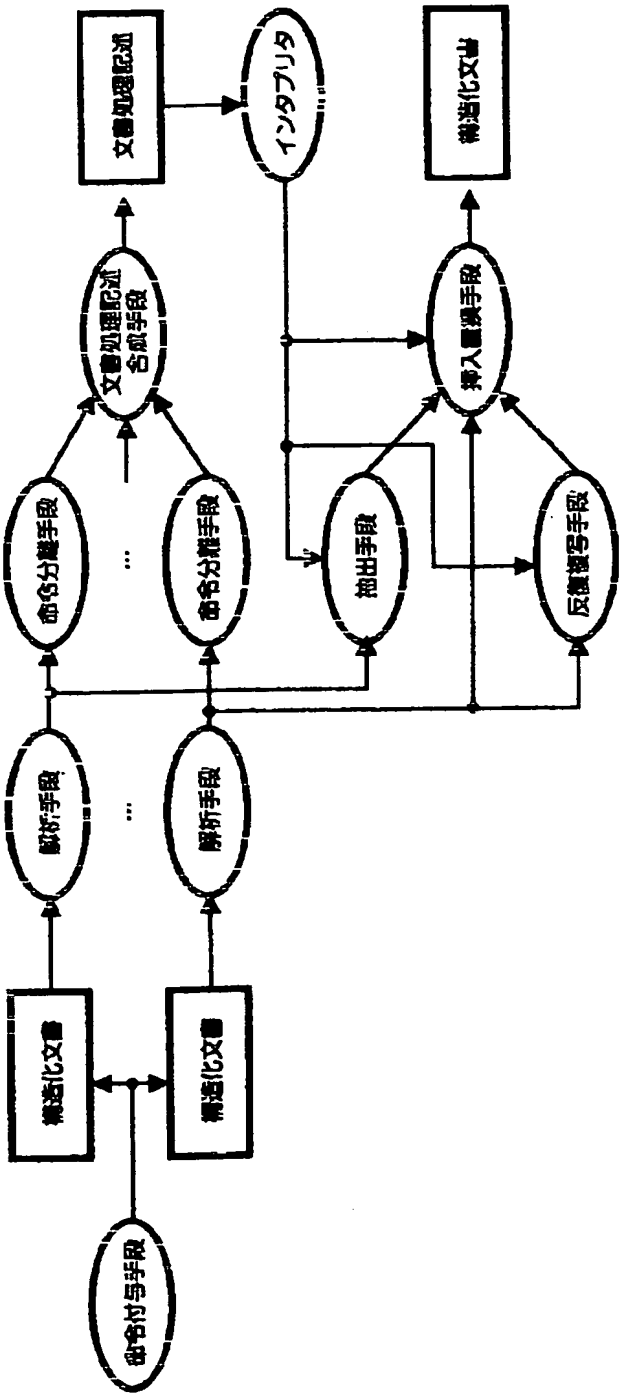
【０３３４】また、例７－Ｂに示すようなタイプの場合には、URLを含む第２のHTTPリクエストを自動的に送信するスクリプト記述（例えばＪａｖａＳｃｒｉｐｔ言語）をserverが出力する（第１の）HTTPレスポンスによって送信されるHTML文書中に挿入することで、人手の介入をすることなく自動的にエラー通知文書を得ることができる。例えば、WWWブラウザ側では、第１のHTTPレスポンスを受信すると、HTML形式の文書に含まれるスクリプトを実行する。そのスクリプトは、前記処理起動記述をserverに送って得られるエラー通知文書を、所定のウィンドウを開いて表示するものとなっている。それ以外の動作は、例７－Ａと略同一であると理解されたい。

【０３３５】要言すれば、本実施例の構成では文書処理の途中で発生したエラーを、文書処理の合成結果とは別にして利用者に通知することができる。したがって、高度で柔軟な文書処理アプリケーションを構成できる。さらに、エラーが発生した時点でエラー通知文書を生成して、文書処理の結果を一時的に格納する保持手段に（エラー通知文書のリクエストに先行して）格納し、後のエラー通知文書の要求に対して出力するように動作することができる。このため、エラーを発生したプロセスとエラーの種類やエラーに関する情報を対応付けて保持するような機構を用意する必要がなくなる。

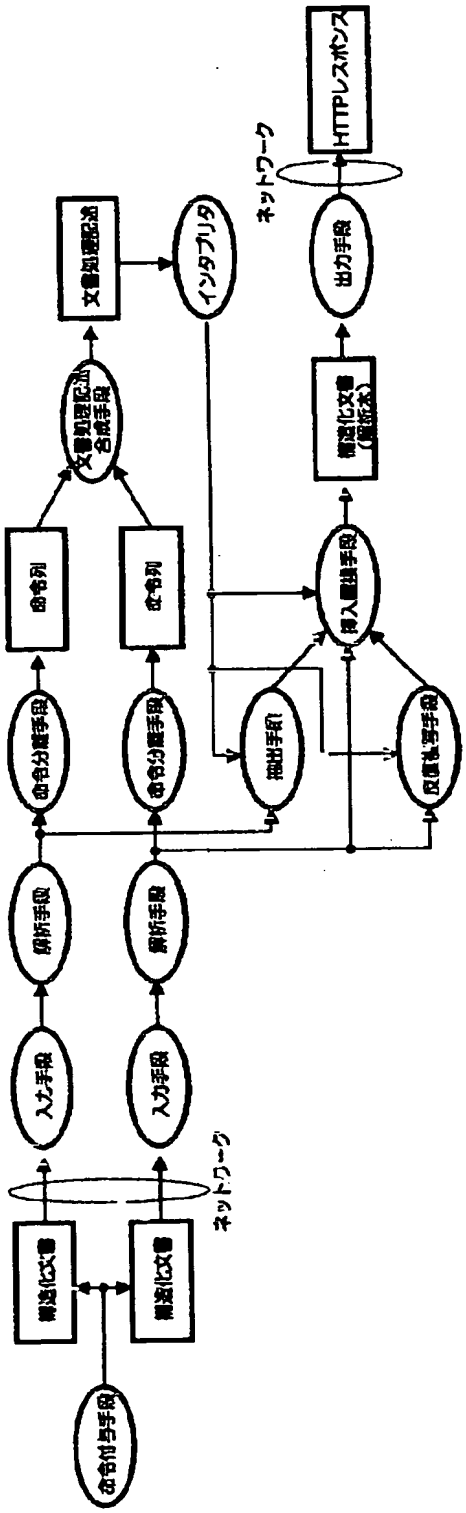
【０３３６】［追補］以上、特定の実施例を参照しながら、本発明について詳解してきた。しかしながら、本発明の要旨を逸脱しない範囲で当業者が該実施例の修正や代用を成し得ることは自明である。すなわち、例示という形態で本発明を開示してきたのであり、限定的に解釈されるべきではない。本発明の要旨を判断するためには、冒頭に記載した特許請求の範囲の欄を参酌すべきである。

【0337】
【発明の効果】以上詳記したように、本発明によれば、文書を章、節、段落、図表、あるいは表題や章題、概要などの複数の素片に分解し各素片をノードとするツリー構造やグラフ構造などのように構造的に表現して取り扱うことができる、優れた構造化文書処理システム及び構造化文書処理方法を提供することができる。
【0338】また、本発明によれば、複数の構造化文書を基にして新たに文書を合成することができる、優れた構造化文書処理システム及び構造化文書処理方法を提供することができる。
【0339】また、本発明によれば、複数の構造化文書から特定の条件を満たす文書部分(「文書部品」)を取り出すとともに文書部品を別の文書中に挿入又は置換して合成を行なうことができる、優れた構造化文書処理システム及び構造化文書処理方法を提供することができる。
【0340】また、本発明によれば、構造化文書から文書部品を抽出したり、各文書部品を雛型文書中に挿入又は置換したりする手続きを記述したスクリプトを用いることなしに構造化文書の合成処理を行うことができる、優れた構造化文書処理システム及び構造化文書処理方法を提供することができる。
【図面の簡単な説明】
【図1】本発明に係る構造化文書処理システムの基本概念を示した機能ブロック図である。
【図2】本発明の第1の実施例に係る構造化文書処理システムの構成を模式的に示した機能ブロック図である。
【図3】第1の原料文書を解析木として表現した図である。
【図4】第2の原料文書を解析木として表現した図である。
【図5】第1の雛型文書を解析木として表現した図である。
【図6】第2の雛型文書を解析木として表現した図である。
【図7】雛型文書1に対して、文書処理記述例1のRP命令で反復複写した結果の文書解析木の例を示した図である。
【図8】雛型文書2に対して、文書処理記述例2のRP命令で反復複写した結果の文書解析木の例を示した図である。
【図9】第1の原料文書と第2の原料文書と第1の雛型文書から合成した文書の解析木を示した図である。
【図10】本発明の第2の実施例に係る構造化文書処理システムの構成を模式的に示した機能ブロック図である。
【図11】第3の原料文書の解析木を示した図である。
【図12】第3の雛型文書の解析木を示した図である。
【図13】本発明の第5の実施例に係る構造化文書処理システムの構成を模式的に示した機能ブロック図である。
【図14】第2の処理起動記述によって起動される処理の流れを示した図である。
【図15】本発明の第6の実施例に係る構造化文書処理システムの構成を模式的に示した機能ブロック図である。
【図16】本発明の第7の実施例に係る構造化文書処理システムの構成を模式的に示した機能ブロック図である。
【図17】本発明の第8の実施例に係る構造化文書処理システムの構成を模式的に示した機能ブロック図である。
【図18】第1の原料文書を標準的なHTMLブラウザで表示した結果を示した図である。
【図19】第2の原料文書を標準的なHTMLブラウザで表示した結果を示した図である。
【図20】第1の原料文書と第2の原料文書と第1の雛型文書から合成して生成されたHTMLファイルを標準的なHTMLブラウザで表示した結果を示した図である。
【図21】第1の原料文書と第2の原料文書と第2の雛型文書から合成して生成されたHTMLファイルを標準的なHTMLブラウザで表示した結果を示した図(第1の実施例)である。
【図22】第3の原料文書を標準的なHTMLブラウザで表示した結果を示した図である。
【図23】第3の原料文書と第3の雛型文書から合成して生成されたHTMLファイルを標準的なHTMLブラウザで表示した結果を示した図(第2の実施例)である。
【図24】原料文書4と原料文書5と原料文書6と雛型文書4から合成されるHTMLファイルを標準的なHTMLブラウザで表示した結果を示した図(第3の実施例)である。
【図25】原料文書4と原料文書5と雛型文書4から合成されるHTMLファイルを標準的なHTMLブラウザで表示した結果を示した図(第4の実施例)である。
【図26】従来技術に係る構造文書処理システムの構成例を示した図である。
【図27】従来技術に係る構造文書処理システムの構成例を示した図である。

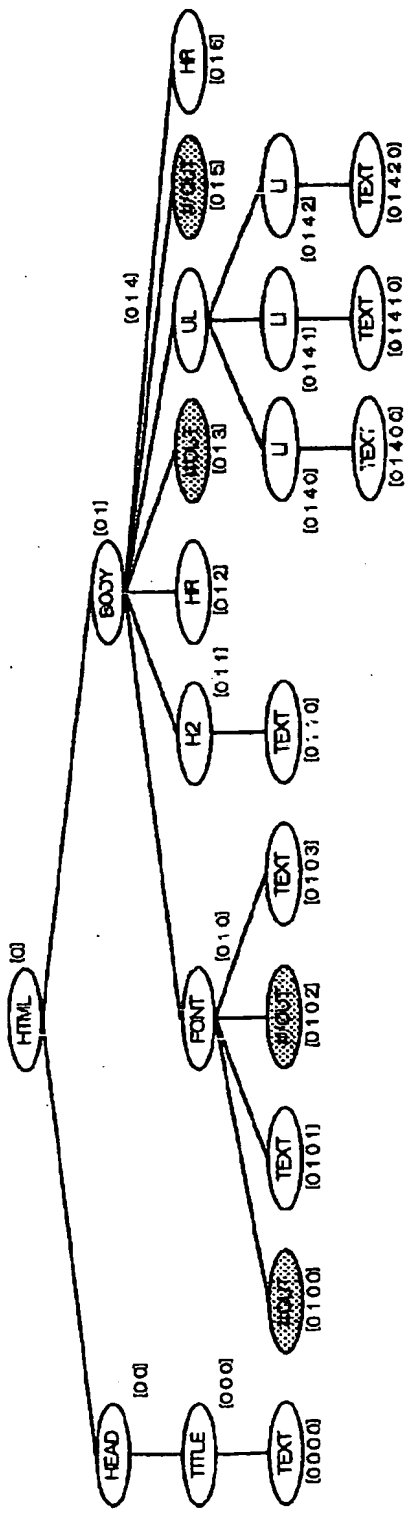
【図1】


文書処理記述
インタプリタ
構造化文書
文書処理記述合成手段
挿入置換手段
命令分離手段
命令分離手段
抽出手段
反復複写手段
解析手段
解析手段
構造化文書
構造化文書
命令付与手段

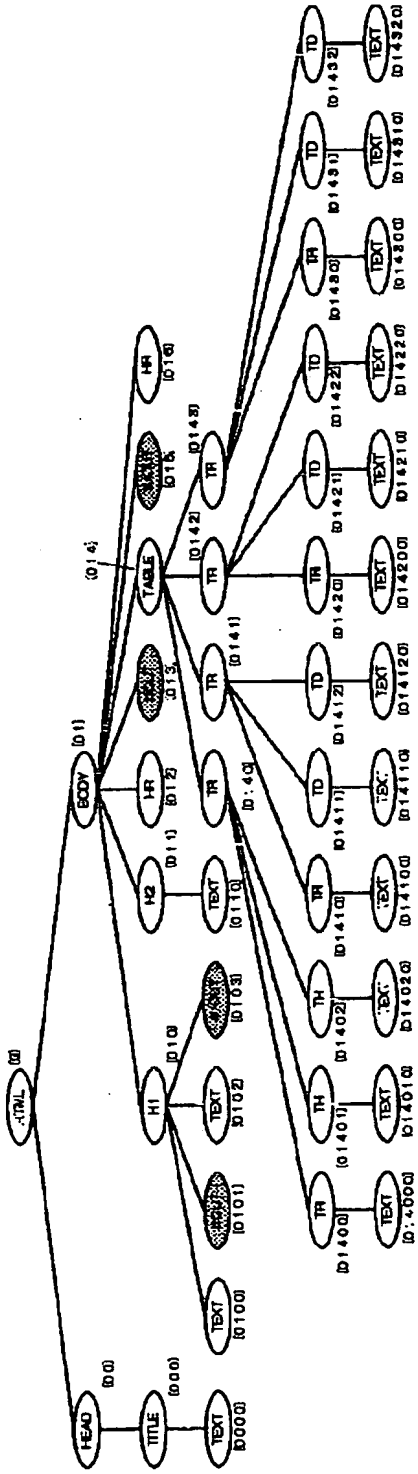
【図2】


HTTPレスポンス
ネットワーク
出力手段
構造化文書(複数本)
文書処理記述
インタプリタ
文書処理記述合成手段
挿入置換手段
命令列
命令列
抽出手段
反復複写手段
命令分離手段
命令分離手段
解析手段
解析手段
入力手段
入力手段
ネットワーク
構造化文書
構造化文書
命令付与手段

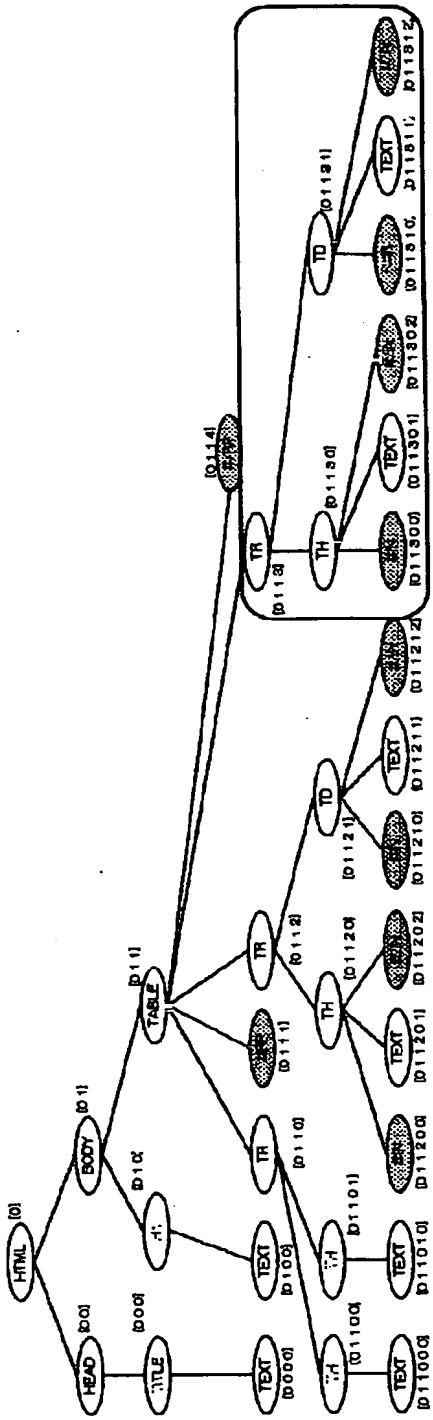
【図3】


HTML
HEAD
TITLE
TEXT
BODY
FONT
H2
HR
UL
LI
TEXT

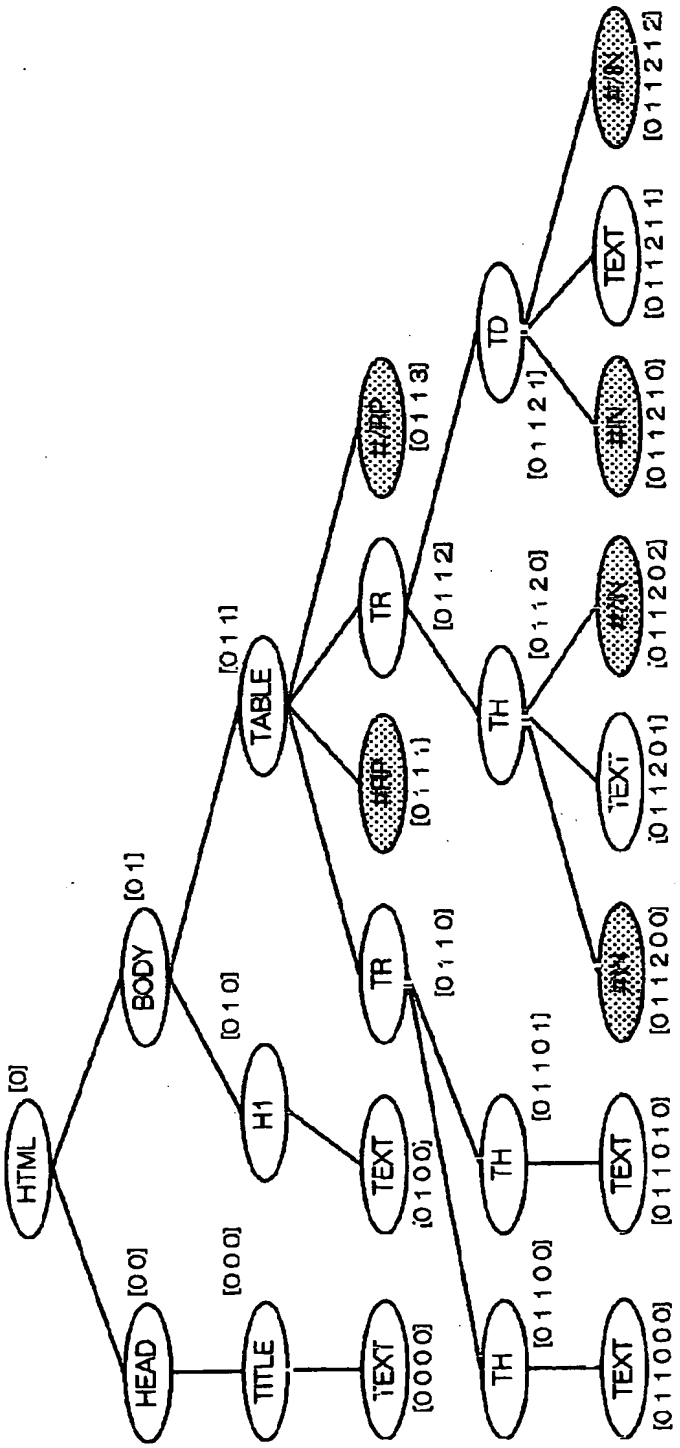
【図4】


HTML
HEAD
TITLE
TEXT
BODY
TABLE
TR
TD
TH
H1
H2
HR
UL
LI

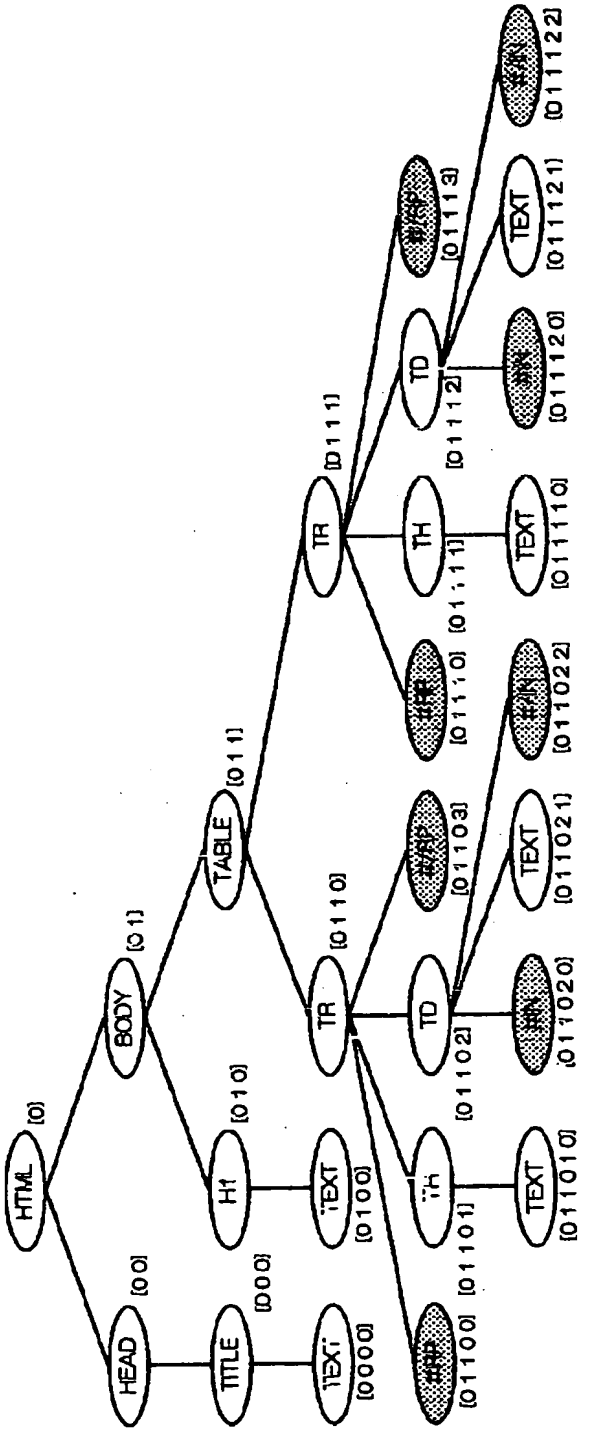
【図7】


HTML
HEAD
TITLE
TEXT
BODY
TABLE
TR
TD
TH

【図5】


HTML [0]
HEAD [00]
BODY [01]
TITLE [000]
H1 [010]
TABLE [011]
TEXT [0000]
TEXT [0100]
TR [0110]
TR [0112]
#TH [0111]
#TH [0113]
TH [01100]
TH [01101]
TH [01120]
TD [01121]
#N [011200]
TEXT [011010]
TEXT [011000]
TEXT [011201]
#N [011202]
TEXT [011210]
TEXT [011211]
#N [011212]

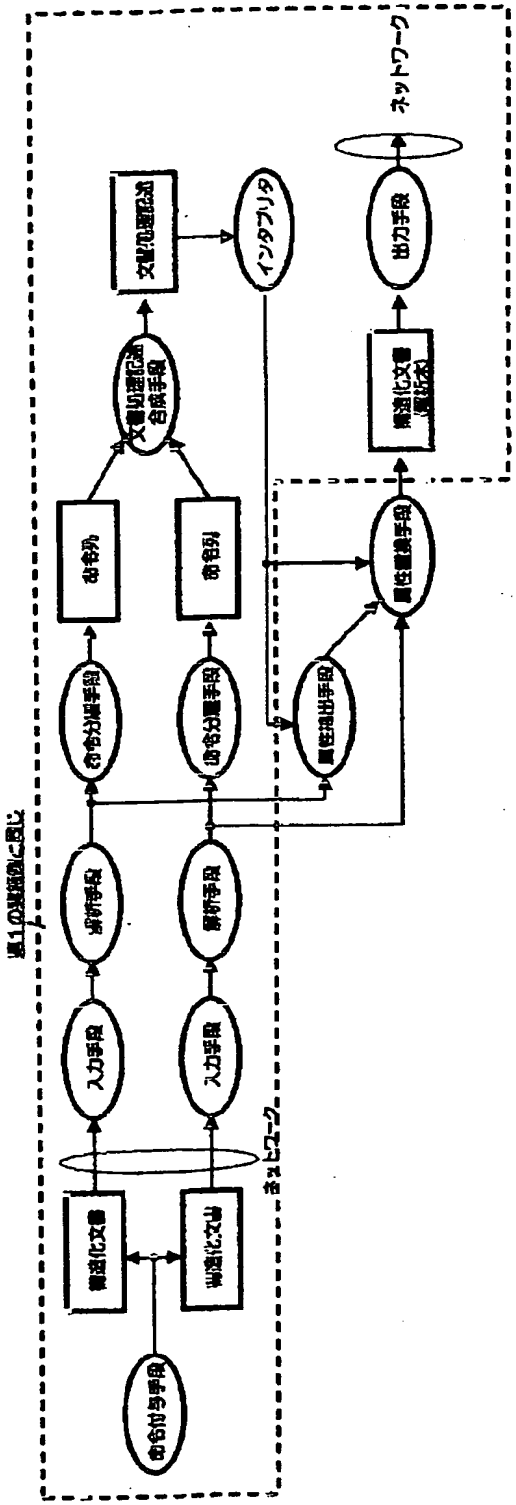
【図6】


HTML [0]
HEAD [00]
BODY [01]
TITLE [000]
H1 [010]
TABLE [011]
TEXT [0000]
TEXT [0100]
TR [0110]
TR [0111]
#TH [01100]
TH [01101]
TD [01102]
#TH [01103]
TD [01112]
#TH [01113]
TEXT [011010]
TH [011020]
TEXT [011021]
#N [011022]
TH [011110]
TEXT [011120]
TEXT [011121]
#N [011122]

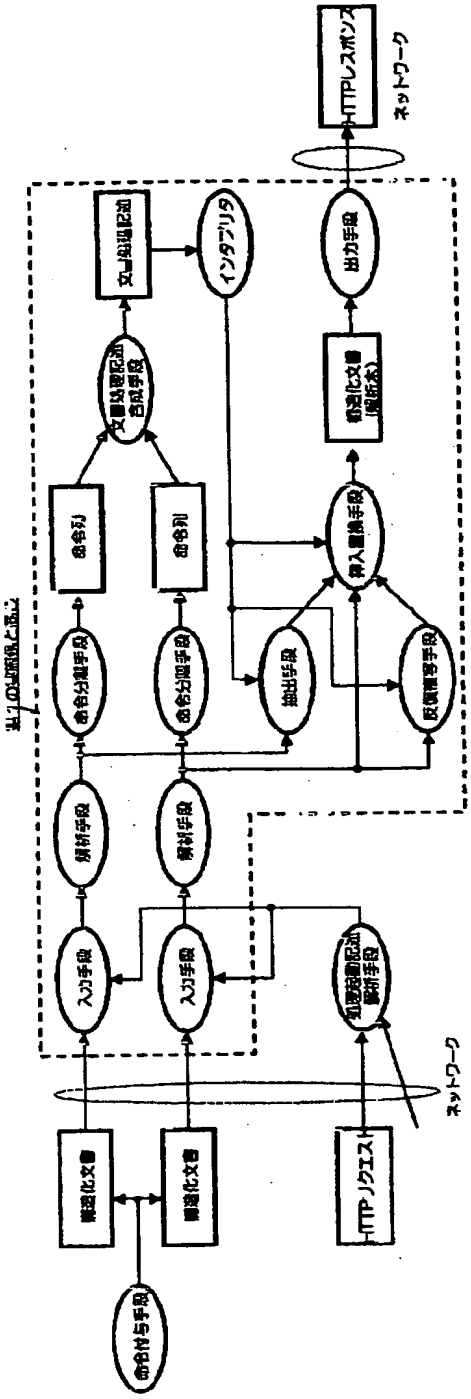
【図8】

【図9】

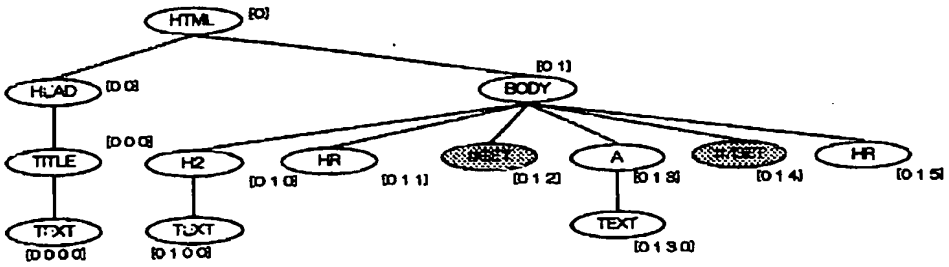
【図10】


ネットワーク
出力手段
インタプリタ
命令列
命令分離手段
解析手段
入力手段
ネットワーク

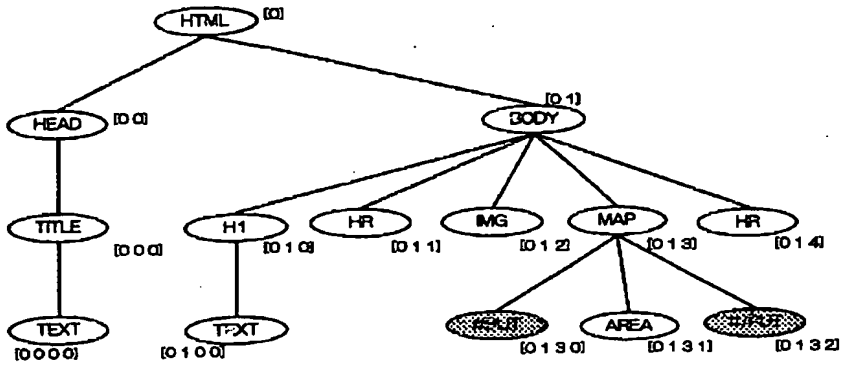
【図13】


ネットワーク
HTTPレスポンス
出力手段
インタプリタ
命令列
命令分離手段
解析手段
入力手段
抽出手段
反復手段
ネットワーク
HTTPリクエスト

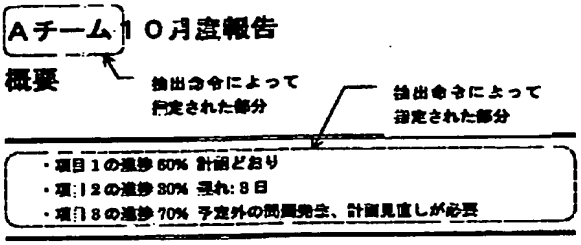
【図11】



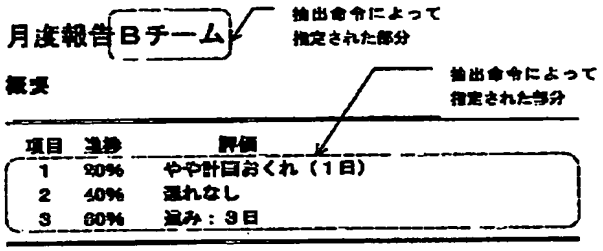
【図12】



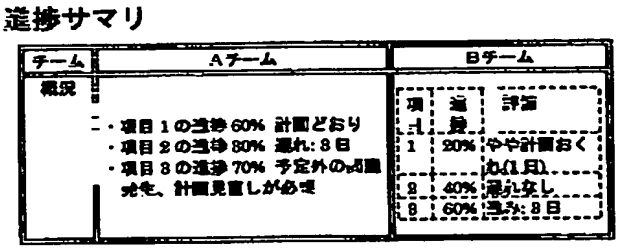
【図18】



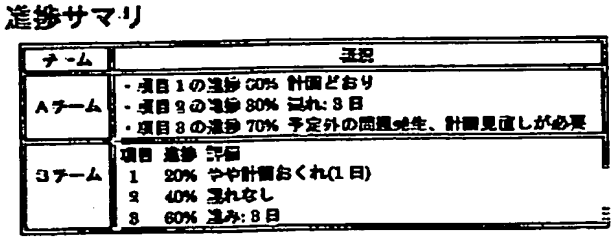
【図19】



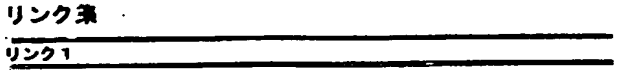
【図21】



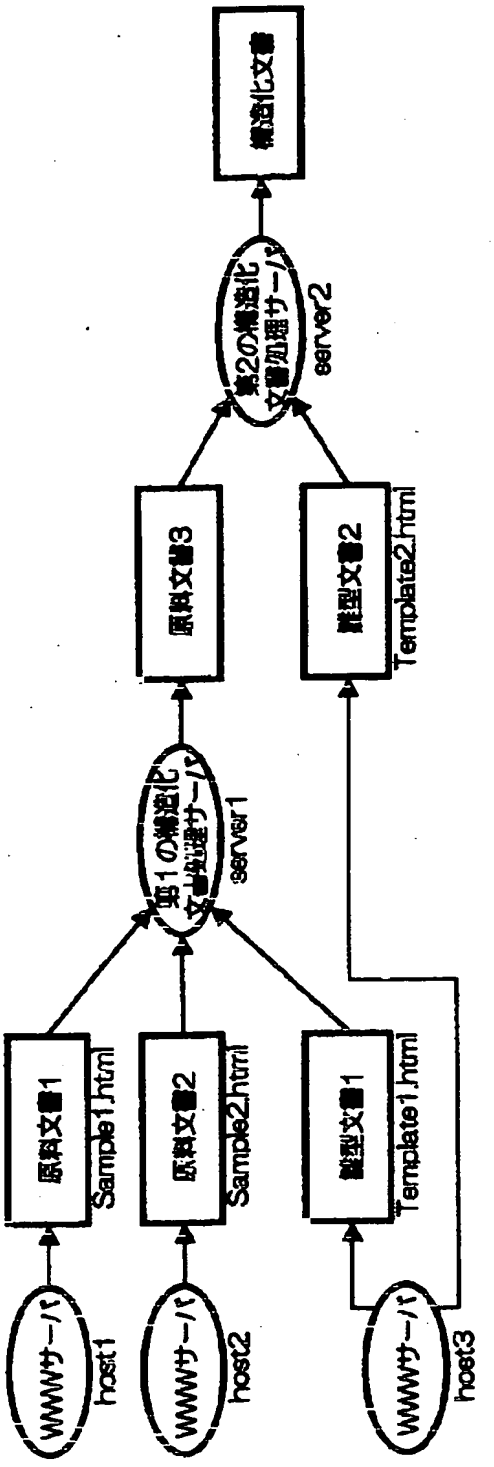
【図20】



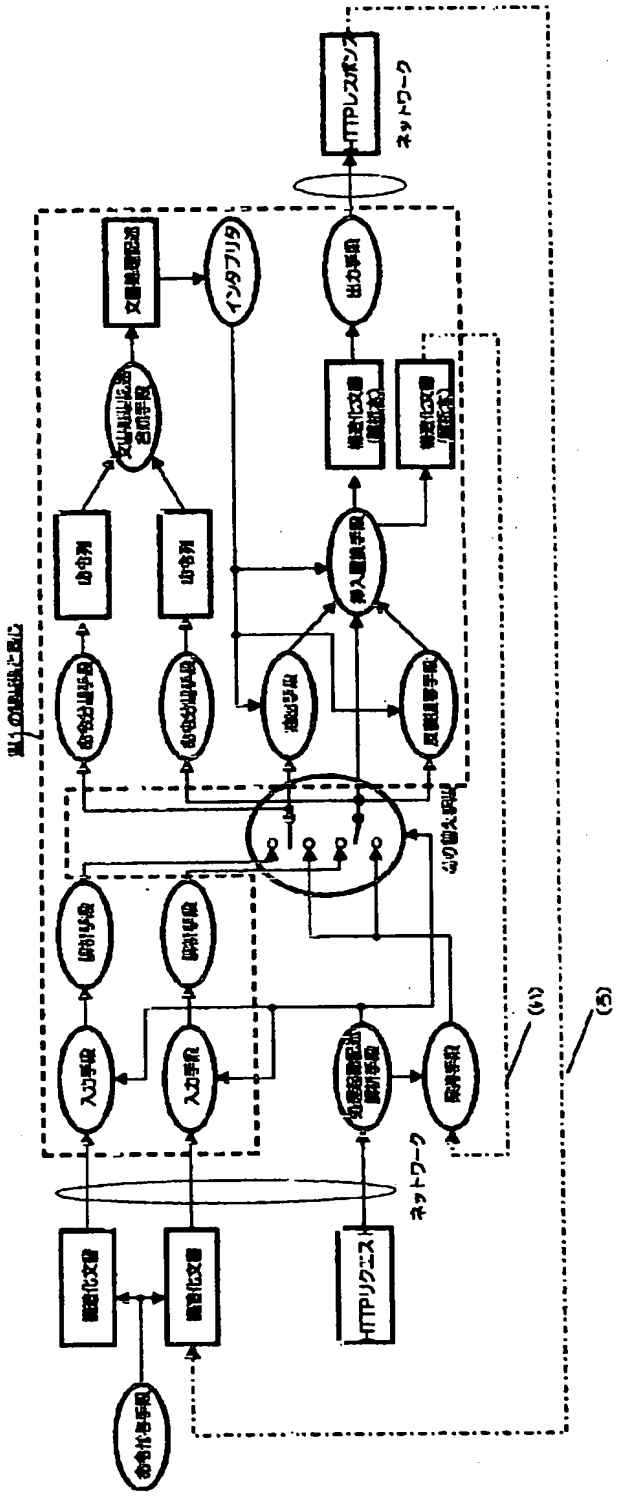
【図22】



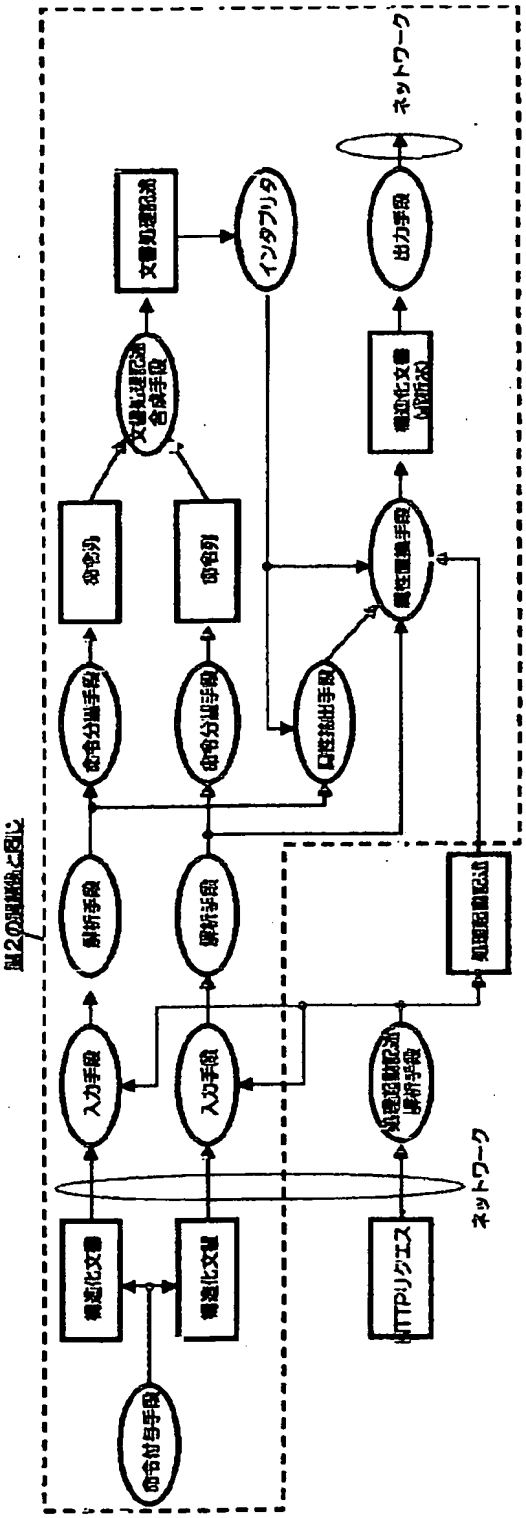
【図14】


構造化文書
第2の構造化文書処理サーバ
server2
原料文書3
雛型文書2
Template2.html
第1の構造化文書処理サーバ
server1
原料文書1
Sample1.html
原料文書2
Sample2.html
雛型文書1
Template1.html
WWWサーバ
host1
WWWサーバ
host2
WWWサーバ
host3

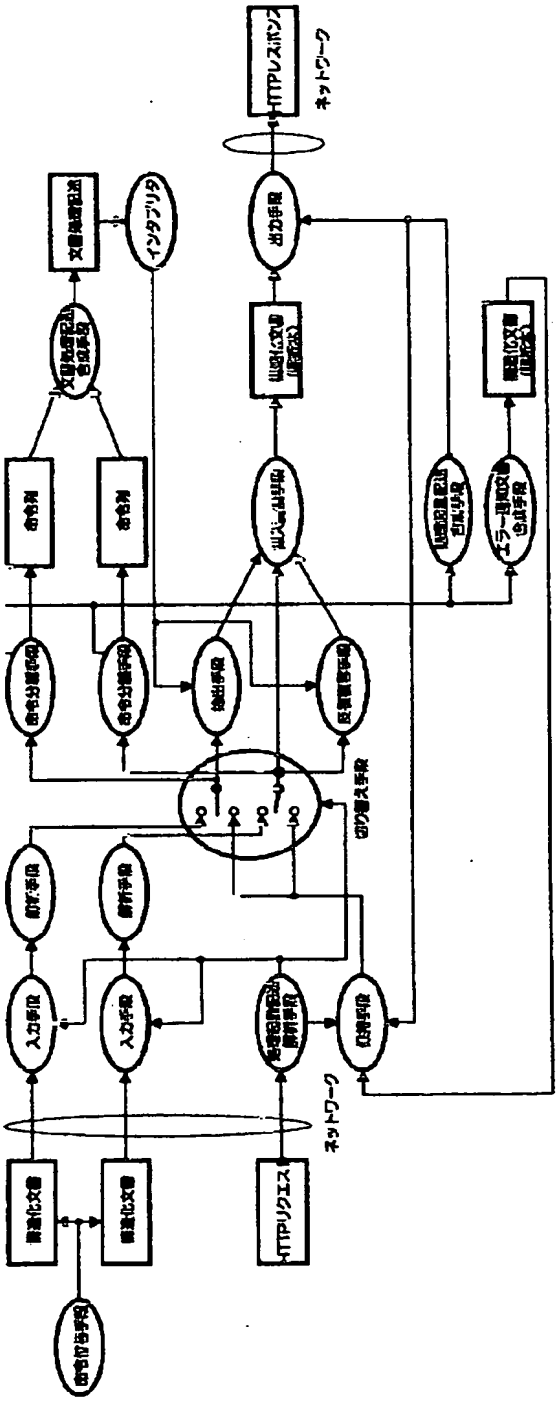
【図15】


HTTPレスポンス
ネットワーク
インタプリタ
出力手段
構造化文書
構造化文書
入力手段
入力手段
HTTPリクエスト
ネットワーク

【図16】


ネットワーク
インタプリタ
出力手段
入力手段
解析手段
命令分離手段
命令列
HTTPリクエスト
構造化文書

【図17】


ネットワーク
インタプリタ
出力手段
入力手段
解析手段
命令分離手段
命令列
HTTPリクエスト
構造化文書

【図２３】

資料集

チャート

【図２４】

進捗サマリ

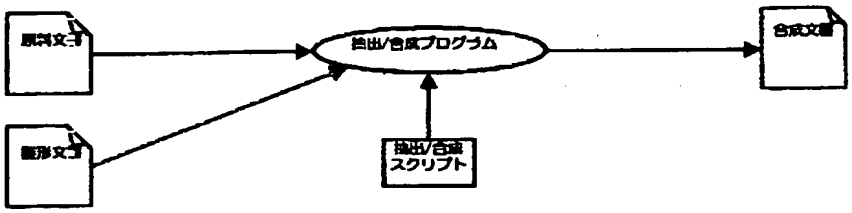
チーム	状況
Aチーム	・項目１の進捗 60% 計画どおり ・項目２の進捗 30% 遅れ: 8 日 ・項目３の進捗 70% 予定外の問題発生、計画見直しが必要
Bチーム	項目 進捗 評価 1 20% やや計画おくれ(1 日) 2 40% 遅れなし 3 60% 進み: 3 日

【図２５】

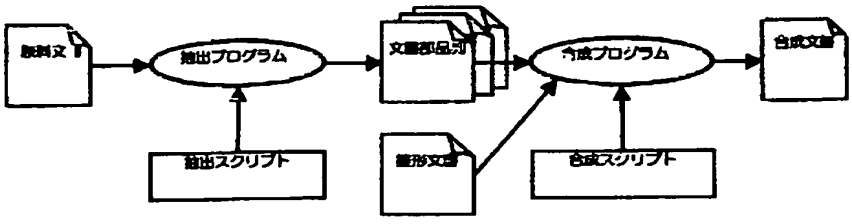
進捗サマリ

チーム	状況
Aチーム	・項目１の進捗 60% 計画どおり ・項目２の進捗 30% 遅れ: 8 日 ・項目３の進捗 70% 予定外の問題発生、計画見直しが必要
Bチーム	項目 進捗 評価 1 20% やや計画おくれ(1 日) 2 40% 遅れなし 3 60% 進み: 3 日

【図２６】



【図２７】



フロントページの続き

(72)発明者　川邉　恵久
神奈川県足柄上郡中井町境430　グリーンテクなかい　富士ゼロックス株式会社内

Fターム(参考)　5B009 NA06 ND04 QA09